Stabilization of a Rotary Wing Unmanned Aerial Vehicle with an Unknown Suspended Payload

Author: A.P. Erasmus

Supervisor: Dr H.W. Jordaan

Co-Supervisor: Mr J. Treurnicht

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at the Faculty of Engineering, Stellenbosch University

March 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2020

Date

Abstract

This thesis addresses the problem of stabilizing a quadrotor with an unknown suspended payload able to swing in one axis. The payload parameters, such as its mass and cable length, are unknown and its swing angle is not available for measurement. The suspended payload significantly alters the flight dynamics of the vehicle as it induces oscillations into the system. This project attempts to design, simulate and practically demonstrate a control strategy to damp these oscillations and maintain stable flight.

Two control strategies were explored, namely a Linear Quadratic Gaussian (LQG) control approach and a robust Model Reference Adaptive Control (MRAC) approach. Both of these control strategies were explored in simulation, of which one was chosen to implement on a practical vehicle. The LQG control approach estimates the payload parameters and its swing angle. Optimal full-state feedback control is implemented to simultaneously control the vehicle and the payload swing angle. The MRAC scheme adapts its controller to change the closed-loop dynamics of the system to that of a predefined reference model. Both strategies were extended to perform well in the presence of external disturbances and sensor noise, of which practical systems suffer. They proved to sufficiently damp the oscillations caused by the payload, but the MRAC scheme has more advantages such that it consists of fewer components and provides consistent performance with different payloads. The MRAC scheme was chosen to implement on a practical vehicle.

A quadrotor vehicle was built to practically demonstrate the effectiveness of the proposed MRAC scheme. The avionics of the vehicle consists of a Pixhawk flight controller running the PX4 flight control stack. The control gains of PX4 were updated according to the custom-built quadrotor and the MRAC scheme was implemented in the flight stack. Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) simulations were performed with the Gazebo simulator to ensure that the implemented MRAC scheme worked as expected in the PX4 environment.

After the success of the simulations, practical flight tests were performed to demonstrate the effectiveness of the MRAC scheme. Three flight tests, each with a different payload, were performed. The MRAC algorithm successfully damped the payload oscillations in each flight and even outperformed a PID controller, tuned for the specific payload, in terms of reference following. MRAC adapted its control parameters to accommodate the specific attached payload in each flight. The flight tests were successful and the algorithm proved to damp the oscillations caused by the payload while maintaining stable flight.

Uitreksel

Die fokus van hierdie tesis is die stabilisering van 'n vierrotor onbemande vliegtuig met 'n onbekende swaai vrag wat in een as kan swaai. Die loonvragparameters, soos die massa en kabellengte, is onbekend en die swaaihoek is nie beskikbaar om te meet nie. Die swaai vrag verander die vlugdinamika van die voertuig aansienlik, aangesien dit ossillasies in die stelsel veroorsaak. Hierdie projek poog om 'n beheerstrategie te ontwerp, te simuleer en prakties uit te voer om hierdie ossillasies te demp en stabiele vlug te handhaaf.

Twee beheerstrategieë word in hierdie tesis ondersoek. Dit is 'n Lineêre Kwadratiese Gaussiese (LKG) benadering en 'n robuuste Model Verwysing Aanpasbare Beheer (MVAB) benadering. Albei beheerstrategieë word in simulasie ondersoek, waarvan een gekies word om op 'n praktiese voertuig te implementeer. Die LKG benadering bereken die loonvragparameters om die swaaihoek af te skat en maak gebruik van optimale terugvoerbeheer om beide die voertuig en die swaaihoek te beheer. Die MVAB skema pas sy beheerder aan om die geslote-lus dinamika van die stelsel te verander na dié van 'n vooraf gedefinieerde verwysingsmodel. Albei strategieë is uitgebrei om goed te presteer in die teenwoordigheid van eksterne versteurings en sensor ruis. Albei strategieë het die ossillasies wat deur die loonvrag veroorsaak word voldoende gedemp, maar die MVAB skema het meer voordele soos dié dat dit minder komponente het en dat dit konsekwente resultate lewer met verskillende loonvragte. Die MVAB skema word daarom gekies om op 'n praktiese voertuig te implementeer.

'n Vierrotor voertuig was gebou om te gebruik vir praktiese vlugtoetse. Die avionika van die voertuig bestaan uit 'n Pixhawk vlugbeheerder wat die PX4 kode hardloop. Die beheer aanwinste van die PX4 beheerargitektuur was aangepas vir die vierroter en die MVAB skema was geïmplementeer in PX4. Beide sagteware-in-die-lus en hardware-in-die-lus simulasies was uitgevoer met die Gazebo simulator om te verseker dat die geïmplementeerde MVAB skema werk in die PX4 omgewing.

Na die sukses van die simulasies, was praktiese vlugtoetse uitgevoer om die MVAB skema prakties te demonstreer. Drie vlugtoetse, elk met 'n ander loonvrag, was uitgevoer. Die MVAB algoritme het die loonvrag ossillasies in elke vlug suksesvol gedemp. Dit het selfs beter gevaar as 'n PID beheerder, ontwerp vir die spesifieke loonvrag, in terme van die volg van die verwysing. MVAB het die beheer parameters aangepas om die spesifieke loonvrag in elke vlug te akkommodeer. Die vlugtoetse was suksesvol en die algoritme het die ossillasies, wat deur die loonvrag veroorsaak word, gedemp terwyl stabiele vlug gehandhaaf was.

Acknowledgements

I would like to thank the following:

- Dr. Willem Jordaan for all his guidance, insight and advice throughout the project.
- Johann Treurnicht for his input regarding the practical quadrotor vehicle.
- Our safety pilot, Michael Basson, for ensuring the safe flight of the quadrotor.
- The technical staff at the E&E Department, including Wessel Croukamp, Johan Arendse, and Wynand van Eeden.
- My friends at ESL for all their support, including Nico Lochner, Ryno Swart, Francois Lombard, Jaak Rademeyer, Henry Kotze, Francois Slabber, Reghard Grobler, and Armand Scholtz.
- My Lord and Saviour Jesus Christ for His Love and strength.
- My fiancé, Willemien Truter, for her love, support, and encouragement.
- My family, Tonie Erasmus, Andra Erasmus and Martinus Erasmus, for all their love and support.

Contents

Abstract ii

Uitreksel iii

Acknowledgements iv

Table of Contents v

List of Figures x

List of Tables xv

List of Abbreviations xvii

1 Introduction 1

1.1 Background . . . 1

1.1.1 Consumer Deliveries . . . 2

1.1.2 Medical Needs . . . 2

1.1.3 Transportation and Delivery Competitions . . . 3

1.1.4 Summary . . . 3

1.2 Project Definition . . . 3

1.3 Thesis Outline . . . 4

2 Literature Study 6

2.1 Multirotor Overview . . . 6

2.2 Multirotor with Payload . . . 7

2.2.1 Grasped Payload . . . 7

2.2.2 Suspended Payload . . . 8

2.3 Multirotor with Unknown Payload . . . 11

2.3.1 Unknown Payload Parameters . . . 11

2.3.2 Unknown Payload States . . . 12

2.4 Summary . . . 12

3 Modelling **14**

3.1 Coordinate Frames . . . 14

3.2 Quaternions . . . 15

3.2.1 Inverse Quaternion . . . 16

3.2.2 Quaternion Multiplication . . . 16

3.3 Six Degrees of Freedom . . . 16

3.3.1 Kinematics . . . 16

3.3.2 Kinetics . . . 17

3.4 Forces and Moments . . . 18

3.4.1 Actuators . . . 18

3.4.2 Gravity . . . 19

3.4.3 Aerodynamics . . . 19

3.4.4 Suspended Payload . . . 20

3.5 Summary . . . 24

4 Hardware Design **25**

4.1 Vehicle Hardware . . . 25

4.1.1 Quadrotor Components . . . 25

4.2 Quadrotor Physical Parameters . . . 29

4.2.1 Mass Moment of Inertia . . . 29

4.2.2 Thrust vs PWM Mapping . . . 30

4.2.3 Motor Time Constant . . . 30

4.2.4 Virtual Yaw Moment Arm . . . 31

4.2.5 Aerodynamic Coefficients . . . 32

4.3 Summary . . . 33

5 Flight Control Toolchain Overview **34**

5.1 PX4 Overview . . . 34

5.1.1 Architecture . . . 34

5.1.2 Estimator . . . 35

5.1.3 Controllers . . . 35

5.1.4 Simulation . . . 35

5.2 PX4 Modifications . . . 35
5.3 Simulation Environment . . . 36
5.3.1 MATLAB/Simulink Simulation . . . 36
5.3.2 Gazebo Simulator . . . 36
5.3.3 Discussion . . . 38
5.4 Practical Flight Setup . . . 38
5.4.1 QGroundControl . . . 38
5.4.2 ROS . . . 39
5.5 Summary . . . 40

6 Vehicle Control System **41**
6.1 Control System Design . . . 41
6.1.1 Control System Design Strategy . . . 41
6.1.2 PX4 Control System Architecture . . . 41
6.1.3 Mixer . . . 42
6.1.4 Angular Rate Controllers . . . 43
6.1.5 Angle Controllers . . . 46
6.1.6 Force and Yaw to Attitude and Thrust Conversion . . . 48
6.1.7 Linear Velocity Controllers . . . 49
6.1.8 Position Controllers . . . 50
6.2 Simulation Results . . . 52
6.3 Practical Flight Test . . . 54
6.4 Summary . . . 55

7 Vehicle with Payload Control System **56**
7.1 Multitrotor and Suspended Payload Plant . . . 57
7.2 Tuned PID Controller Design . . . 60
7.3 Estimation and Full-State Feedback Approach . . . 61
7.3.1 Payload Mass Estimation . . . 62
7.3.2 Cable Length Estimation . . . 63
7.3.3 EKF for the Payload Swing Angle Estimation . . . 65
7.3.4 LQR Control . . . 67
7.3.5 Summary . . . 70

7.4 Adaptive Control Approach . . . 71
7.4.1 Control Law . . . 72
7.4.2 Adaptive Law . . . 75
7.4.3 Robustness . . . 78
7.4.4 Summary . . . 82
7.5 Comparison and Summary . . . 83

8 Practical Implementation and Results **84**
8.1 PX4 Implementation . . . 84
8.2 PX4-Gazebo Simulation Results . . . 85
8.2.1 Software-in-the-Loop . . . 85
8.2.2 Hardware-in-the-Loop . . . 86
8.3 Practical Flight Tests . . . 87
8.3.1 Flight Results . . . 88
8.3.2 Out-of-Plane Oscillations . . . 90
8.4 Summary . . . 94

9 Conclusion **95**
9.1 Current Solutions in Literature . . . 95
9.2 Proposed Solution . . . 95
9.2.1 LQG Controller . . . 96
9.2.2 Adaptive Controller . . . 96
9.3 Practical Flight Tests . . . 96
9.4 Future Work . . . 97

References **102**

A Quadrotor Physical Parameter Calculations **103**
A.1 Propulsion System Performance . . . 103
A.2 Calculations of the Quadrotor Payload Capabilities . . . 103
A.3 Mass Moment of Inertia Experiment . . . 104
A.4 Normalized Motor Thrust vs PWM Mapping . . . 106
A.5 Calculation of the Virtual Yaw Moment Arm . . . 106
A.5.1 Rotor Drag Point Force Method . . . 106

A.5.2 Blade Element Theory 107
A.5.3 Experimental Method 109

B Quadrotor Control System Design **110**
B.1 Calculations of the Force to Attitude and Thrust Conversion 110
B.2 Linear Plant of the Longitudinal Velocity 111
B.3 Lateral, Heave and Directional Controller Design 112
B.4 Control System Gains 112
B.4.1 Attitude Controllers 112
B.4.2 Translational Controllers 115

List of Figures

1.1 A VTOL [2], fixed-wing UAV [3] and RUAV [4]. . . . 1
1.2 A multirotor used for irrigation [5]. . . . 1
1.3 The Amazon Prime Air vehicle [9]. . . . 2
1.4 The Alphabet Wing UAV [10]. . . . 2
1.5 A quadrotor carrying an suspended payload. . . . 3

2.1 Illustration of a quadrotor. . . . 6
2.2 A roll, pitch and yaw maneuver. . . . 7
2.3 Example of a multirotor with a grasped payload [15]. . . . 8
2.4 Example of a multirotor with a suspended payload [16]. . . . 8
2.5 Input shaping with deconstructive superposition. . . . 9
2.6 Block diagram of hybrid control system for multirotor with suspended payload. . . . 10
2.7 Multirotor with a suspended payload. . . . 11
2.8 Summary of literature study on multirotors with payloads . . . 13

3.1 The inertial and body frames. . . . 14
3.2 Axis Angle Rotation. . . . 15
3.3 2D Quadrotor and payload model. . . . 20
3.4 Pendulum model. . . . 21
3.5 Quadrotor and payload model block diagram. . . . 24

4.1 Diagram of the different components of a quadrotor. . . . 25
4.2 Motor current measure module. . . . 26
4.3 Payload angle measure module. . . . 26
4.4 Open-source flight control software stacks [36, 37, 38, 39]. . . . 27
4.5 Custom built quadrotor with a suspended payload. . . . 28
4.6 Simplified quadrotor model . . . 29
4.7 The raw PWM and thrust relation of each motor. . . . 31
4.8 The normalized and fitted PWM and thrust relation of each motor. . . . 31

4.9 A Motor Step Response . . . 31
4.10 Qaudrotor flying at a constant velocity. . . . 32

5.1 PX4 building blocks. . . . 34
5.2 Block diagrams of the simulation environments. . . . 36
5.3 Block diagram of the practical setup. . . . 36
5.4 PX4 SIL Gazebo communication . . . 37
5.5 PX4 HIL Gazebo communication . . . 37
5.6 Custom built quadrotor Gazebo model. . . . 37
5.7 Practical flight setup illustration. . . . 38
5.8 ROS communication setup . . . 39
5.9 ROS simulation architecture diagram . . . 40
5.10 ROS practical flight architecture diagram . . . 40

6.1 PX4 control system architecture. . . . 42
6.2 The implemented PX4 angular rate controller. . . . 44
6.3 Pitch rate controller design block diagram. . . . 45
6.4 Root locus of the pitch rate dynamics. . . . 45
6.5 Root locus of the pitch rate controller. . . . 45
6.6 Step response of the pitch rate controller. . . . 46
6.7 Disturbance rejection of the pitch rate controller. . . . 46
6.8 The implemented PX4 angle controller. . . . 47
6.9 Pitch angle controller design block diagram. . . . 47
6.10 Root locus of the pitch dynamics. . . . 48
6.11 Root locus of the pitch controller. . . . 48
6.12 Step response of the pitch controller. . . . 48
6.13 Longitudinal velocity controller design block diagram. . . . 49
6.14 Root locus of the longitudinal velocity dynamics. . . . 50
6.15 Root locus of the longitudinal velocity controller. . . . 50
6.16 Step response of the velocity controller. . . . 50
6.17 Disturbance rejection of the velocity controller. . . . 50
6.18 The implemented PX4 position controller. . . . 51
6.19 North position controller design block diagram. . . . 51

6.20 Root locus of the north position dynamics. 51
6.21 Root locus of the north position controller. 51
6.22 Step response of the north position controller. 52
6.23 Non-linear simulation step response of the pitch rate. 53
6.24 Non-linear simulation step response of the pitch angle. 53
6.25 Non-linear simulation step response of the longitudinal velocity. 53
6.26 Non-linear simulation step response of the north position. 53
6.27 The SIL longitudinal position response under the influence of sensor noise. . . . 53
6.28 The SIL longitudinal velocity response under the influence of sensor noise. . . . 53
6.29 Practical flight test quadrotor. 54
6.30 Flight test north position response. 55
6.31 Flight test longitudinal velocity response. 55

7.1 The longitudinal velocity of the quadrotor after a position step input. 56
7.2 The quadrotor and suspended payload system. 57
7.3 Root locus of the longitudinal dynamics of the quadrotor with a suspended payload. 58
7.4 The longitudinal velocity dynamics of the quadrotor with suspended payloads of different masses. 59
7.5 The longitudinal velocity dynamics of the quadrotor with suspended payloads of different cable lengths. 59
7.6 Root locus of the velocity dynamics of a quadrotor with a payload. 60
7.7 Root locus of the tuned PID longitudinal velocity controller. 60
7.8 Step response of the tuned longitudinal velocity controller. 61
7.9 The RLS estimates of the payload mass. 63
7.10 The single-sided amplitude spectrum of the north velocity. 64
7.11 The estimated rod length for various payloads. 64
7.12 The estimated swing angle produced by the EKF. 66
7.13 The estimated swing angle produced by the EKF with a 30% cable length error. 67
7.14 The quadrotor's north velocity with PID and LQR control. 68
7.15 The quadrotor's longitudinal velocity with LQR control in the presence of a disturbance. 69
7.16 The quadrotor's longitudinal velocity with LQR control in the presence of sensor noise. 69
7.17 The effect of aerodynamic drag on the payload swing angle. 69

7.18 Comparison of PID and LQR control with aerodynamic drag. 70
7.19 Steady-state LQG swing angle estimate with aerodynamic drag. 70
7.20 Block diagram of the MRAC architecture. 71
7.21 General feedback block diagram of the MRAC control law. 73
7.22 The longitudinal velocity response of the linear plant with the MRAC control law. 74
7.23 Pole-zero plot of the closed-loop system with $\omega_\lambda = 5$ rad/s. 74
7.24 Pole-zero plot of the closed-loop system with $\omega_\lambda = 50$ rad/s. 75
7.25 The longitudinal velocity response of the non-linear model with the MRAC control law. 75
7.26 The MRAC longtudinal velocity response of the non-linear quadrotor model. . . 77
7.27 The change in the adaptive parameters from their initial values. 77
7.28 Pole-zero plot of the closed-loop system with changing control parameters at different timestamps. 78
7.29 Continuous switching-σ function. 79
7.30 MRAC without leakage. 80
7.31 MRAC with leakage. 80
7.32 The modified MRAC longitudinal velocity response with sensor noise and a disturbance at $t = 40$ s. 82
7.33 The change in the adaptive parameters and u_a from their initial values. 82

8.1 Root locus of the updated north position controller. 85
8.2 Step response of the updated north position controller. 85
8.3 PX4-Gazebo simulation results of the north position response. 85
8.4 PX4-Gazebo simulation results of the longitudinal velocity response. 86
8.5 The change in the adaptive parameters from their initial values during the PX4-Gazebo simulation. 86
8.6 The CPU load and RAM usage of the Pixhawk during a HIL simulation. 87
8.7 Practical flight test of the quadrotor with a suspended payload. 87
8.8 The practical response of the tuned PID controller with a 1 kg payload. 88
8.9 The practical response of the MRAC scheme with a 1 kg payload. 88
8.10 The measured payload angle of the practical flight with the 1 kg payload. 89
8.11 The change in the adaptive parameters from their initial values of the practical flight with the 1 kg payload. 89
8.12 Comparison of the practical response for different payloads. 90

8.13 The rod without any flex during a practical flight test. 90
8.14 The flex of the rod during a practical flight test. 90
8.15 The practical lateral velocity response with different payloads. 91
8.16 The FFT of the practical lateral velocity response with different payloads. . . . 91
8.17 A quadrotor carrying a suspended payload swinging in both directions. 92
8.18 The PX4-Gazebo position simulation result with the lateral MRAC algorithm. . 92
8.19 The PX4-Gazebo velocity simulation result with the lateral MRAC algorithm. . 92
8.20 The practical response with the lateral MRAC scheme. 93
8.21 Illustration of payload rod flex during a roll maneuver. 93

A.1 Inertia experiment setup. 104
A.2 Inertia experiment perturbation. 104
A.3 Point drag force acting on a propeller. 106
A.4 Rotor airfoil illustration. 107

B.1 Illustration of the commanded inertial force. 110
B.2 Illustration of the inertial linear velocity dynamics of a quadrotor. 111

List of Tables

4.1 The function of each of the quadrotor's components. . . . 26
4.2 Designed quadrotor components. . . . 28
4.3 Quadrotor payload capabilities. . . . 28
4.4 Mass moment of inertia mathematical results 29
4.5 Mass moment of inertia experimental results 30
4.6 Calculated values of the virtual yaw moment arm coefficient. . . . 32
4.7 Physical parameter values of the quadrotor 33

5.1 Summary of the toolchain components. . . . 40

6.1 Standard PID control improvements. . . . 44

7.1 Comparison of the LQG approach and the MRAC approach. . . . 83

A.1 T-Motor MN5212 340KV performance. . . . 103
A.2 Mass Moment of Inertia Experiment Results 105
A.3 Virtual yaw moment arm experimental calculations. . . . 109

B.1 The IMU filter parameters. . . . 113
B.2 The pitch rate controller gains and corresponding parameters. . . . 113
B.3 The pitch angle controller gains and corresponding parameters. . . . 113
B.4 The roll rate controller gains and corresponding parameters. . . . 114
B.5 The roll angle controller gains and corresponding parameters. . . . 114
B.6 The yaw rate controller gains and corresponding parameters. . . . 114
B.7 The yaw angle controller gains and corresponding parameters. . . . 114
B.8 The longitudinal velocity controller gains and corresponding parameters. . . . 115
B.9 The north position controller gains and corresponding parameters. . . . 115
B.10 The lateral velocity controller gains and corresponding parameters. . . . 115
B.11 The east position controller gains and corresponding parameters. . . . 116
B.12 The downward velocity controller gains and corresponding parameters. . . . 116

B.13 The downward position controller gains and corresponding parameters. 116

List of Abbreviations

$\mathscr{H}_\infty$ H-Infinity

2D 2-Dimensional

3D 3-Dimensional

ARC Adaptive Robust Control

CoM Center of Mass

CPU Central Processing Unit

DCM Direct Cosine Matrix

EKF Extended Kalman Filter

ESC Electronic Speed Controller

FAA Federal Aviation Administration

FFT Fast Fourier Transform

GPS Global Positioning System

HIL Hardware-in-the-Loop

IMU Inertial Measurement Unit

LiDAR Light Detection and Ranging

LiPo Lithium Polymer

LPF Low Pass Filter

LQG Linear Quadratic Gaussian

LQR Linear Quadratic Regulator

MRAC Model Reference Adaptive Control

NASA National Aeronautics and Space Administration

NED North-East-Down

PCB Printed Circuit Board

PD Proportional Derivative

PID Proportional Integral Derivative

PWM Pulse Width Modulation

QGC QGroundControl

RAM Random Access Memory

RCAC Retrospective Cost Adaptive Control

RISE Robust Integral of the Sign of the Error

RLS Recursive Least Squares

ROS Robot Operating System

RTOS Real Time Operating System

RUAV Rotary-Wing UAV

SDF Simulation Description Format

SDRAM Synchronous Dynamic RAM

SIL Software-in-the-Loop

SISO Single-Input-Single-Output

SPR Strictly Positive Real

TPA Throttle PID Attenuation

UAV Unmanned Aerial Vehicle

UTM Unmanned Aircraft Systems Traffic Management

VTOL Vertical Take-off and Land

XML Extensible Markup Language

1. Introduction

1.1 Background

The Unmanned Aerial Vehicle (UAV) sector of the aviation industry is growing and under extensive research and development. These vehicles include multirotors, formally known as Rotary-Wing UAVs (RUAVs), fixed-wing UAVs and Vertical Take-off and Land (VTOL) vehicles, shown in Fig. 1.1. The applications of these vehicles include surveillance, security, aerial photography, delivery, and 3D mapping.

Multiple industries are starting to realize the applications for these vehicles. They are currently being used in agriculture, mining, filming, and delivery to name but a few. UAVs are used in agriculture to monitor crops for data analysis to be able to understand how to yield a larger produce. They are even used for irrigation, as shown in Fig. 1.2. The mining industry is making use of UAVs to gather geospatial data to better manage its operations. To enable more industries to incorporate these vehicles into their workflow, larger systems need to be put into place to ensure the safe use of these vehicles. In 2019, National Aeronautics and Space Administration (NASA) is working with the Federal Aviation Administration (FAA) on an Unmanned Aircraft Systems Traffic Management (UTM) platform to overcome the challenges of flying UAVs in general air space [1]. Therefore, the use of UAV platforms to accomplish the above-mentioned tasks will soon become even more popular.

Figure 1.1: A VTOL [2], fixed-wing UAV [3] and RUAV [4].

Figure 1.2: A multirotor used for irrigation [5].

These applications of UAVs have one thing in common, which is that the vehicle is carrying some kind of payload. The payload can vary from a camera, a Light Detection and Ranging (LiDAR) sensor or even a medical package for delivery. Most of these applications make use of a static payload where the same payload is used for each flight, such as a camera. Therefore, the control system of the vehicle can be optimized for the specific payload. However, in the context of transportation, the vehicle will carry a different payload with each flight. Therefore, the control system needs to accommodate a wide range of different payloads. This is the focus of this project.

The applications of using UAVs for transport include consumer deliveries, military use, medical or emergency needs, and search and rescue. Companies and research groups are already investing resources into this market and investigating the potential of using UAVs for this purpose.

1.1.1 Consumer Deliveries

Consumer deliveries with UAVs are becoming a reality with companies such as Flirtey, Alphabet, and Amazon investigating the potential.

Flirtey performed the first FAA approved drone delivery in 2016 in the United States [6]. Flirtey's multirotor transports the package to the desired location and lowers the package, attached to a tether, to the ground. They are currently delivering Domino's pizza in New Zealand. Amazon's drone delivery service is called Amazon Prime Air, shown in Fig. 1.3, and made its first delivery in 2016 [7]. Their multirotor delivers the package by landing at the desired location, dropping the package, taking off and returning to the warehouse. Amazon is currently running a preliminary service in England, delivering packages to clients' homes. Alphabet's drone delivery service is called Wing and they make use of a VTOL vehicle, as shown in Fig. 1.4 [8]. They are currently delivering food and coffee in Australia by using a tether to lower the package at the client's home.

All these projects showcase the need for such services, which is fast delivery to your doorstep. The advantage of using UAVs is that they can travel shorter distances to their destination and skip any form of traffic, which results in faster arrival times. They can also fly to hard to reach locations, making it an attractive solution in rural areas.

Figure 1.3: The Amazon Prime Air vehicle [9].

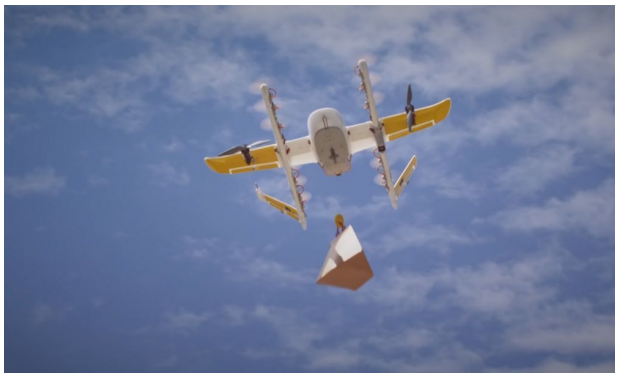


Figure 1.4: The Alphabet Wing UAV [10].

1.1.2 Medical Needs

Another application for using UAVs for transportation is for medical needs. A multirotor was used to deliver an organ for a transplant for the first time, in April 2019 [11]. This was a big breakthrough as the success of this mission showcased the benefits of using UAVs for tasks such as these. The risk of the organ arriving in a bad condition increases with every passing second in transport. Usually, the organs are transported by airlines which introduce large delays in the transportation system. By making use of an UAV, this delay is greatly reduced. The use of an UAV also has the added benefit for serving rural areas that could not otherwise be reached by conventional transport.

1.1.3 Transportation and Delivery Competitions

The development and research of UAVs used for transport and delivery are encouraged by annually held competitions such as the Lake Victoria Challenge and the UAV Challenge: Medical Rescue. Participants are required to do a predefined flight with certain capabilities such as transporting a payload with a given mass across a predefined route. These competitions encourage the participants to develop new strategies and push the limits of what UAVs can execute.

1.1.4 Summary

It is clear from these projects that there is a big need to make use of UAVs for transportation. This is still a growing field and is starting to become increasingly popular.

The above-mentioned projects do not come without their shortcomings and constraints. The payload that the vehicle carries for delivery is unknown before a flight and can have a significant effect on the flight dynamics of the vehicle. The payloads that the industry UAVs carry are restricted to a certain size and mass, such that they have a negligable effect on the flight dynamics of the vehicle. To truly transport an unknown payload, the UAV needs to adapt to the specific payload attached. Therefore, an adaptive strategy for the flight controllers is needed. This project attempts to accomplish this.

1.2 Project Definition

This project aims to design, implement and practically demonstrate a quadrotor transporting an unknown suspended payload in stable flight. The payload, shown in Fig. 1.5, is unknown as its parameters, m_p, l, k and c, are not known and its state, β, is not directly available for measurement.

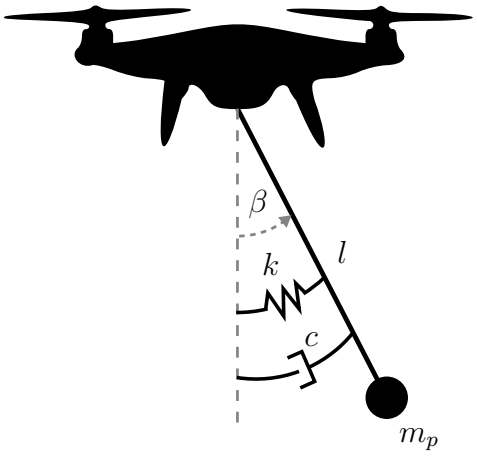


Figure 1.5: A quadrotor carrying an suspended payload.

The suspended payload has a significant effect on the flight characteristics of the quadrotor vehicle as it induces oscillations into the system. This project aims to damp these oscillations to ensure stable flight.

It is assumed that the payload is attached to the Center of Mass (CoM) of the vehicle. Therefore, the payload does not affect the attitude of the vehicle but only its linear velocity and position. For this project, the payload will only be able to swing in one axis. The principle of superposition can then be applied to duplicate the proposed solution in the other axis to handle payloads that can swing in both directions.

The problem is solved by breaking it down to the following steps:

1. Identify the current solutions of multirotors transporting payloads.
2. Derive a mathematical model of a quadrotor and suspended payload system to be used in simulations.
3. Identify the current state-of-the-art flight controllers and flight control stacks, suitable for the control system modifications needed.
4. Build a physical quadrotor with a suspended payload, for the practical experiment.
5. Identify the required software tools needed for simulation and practical flights.
6. Design, simulate and practically demonstrate a control system for the built quadrotor, without a payload.
7. Explore two different control strategies in simulation to ensure stable flight of the quadrotor and payload system.
8. Implement one of the control strategies and practically demonstrate the solution in a practical flight test.

These steps describe the process to achieve the main outcome of this project, which is to practically demonstrate a solution capable of ensuring stable flight of a quadrotor with an unknown suspended payload.

1.3 Thesis Outline

The layout of the thesis is presented in this section.

Chapter 2 contains a literature study, presenting the current solutions to the problem found in literature. The focus of this project is compared to the current solutions.

Chapter 3 derives a mathematical model of the quadrotor and suspended payload model. The derived differential equations can be used to simulate the system and design the relevant controllers.

Chapter 4 describes the design of the custom-built practical quadrotor vehicle. The necessary parameters of the quadrotor are obtained to simulate the practical vehicle.

Chapter 5 identifies the needed software tools such as the flight control firmware, the ground control station software, and the simulation environment. The purpose of each tool and its function in the larger context of this project is presented.

Chapter 6 presents the design, simulation and practical demonstration of the flight control system for the quadrotor vehicle without the suspended payload. This is to ensure that the controllers work as expected, the vehicle can maintain stable flight and the simulation environment is a good representation of the physical system.

Chapter 7 identifies and explores two control strategies that attempt to damp the oscillations caused by the payload. These strategies are explored in simulation and compared to one another. Thereafter, one strategy is chosen to implement and demonstrate on the practical vehicle.

Chapter 8 describes the implementation and simulation of the proposed solution. Practical flight tests are then performed to demonstrate the effectiveness of the strategy. The practical results are presented and observations and conclusions are discussed.

Chapter 9 draws some conclusions of the project and presents possible future work.

2. Literature Study

This chapter is a literature study on the stabilization of multirotors used for transporting payloads. In the first section, a brief overview of multirotors and the main types of flight control systems are given. Thereafter, the different types of payloads and different control schemes are explored. Lastly, a study on the literature available on using multirotors for transporting unknown payloads is presented. This chapter concludes with a summary of the literature discussed and compares it to the focus of this project.

2.1 Multirotor Overview

A multirotor vehicle consists of multiple variable speed motor-propellor pairs. The name of the multirotor usually indicates the number of motors, e.g. a quadrotor has four motors and an octarotor has eight motors. It has six degrees of freedom and performs maneuvers by running different combinations of motors at different speeds. Consider the plus-configuration quadrotor shown in Fig. 2.1.

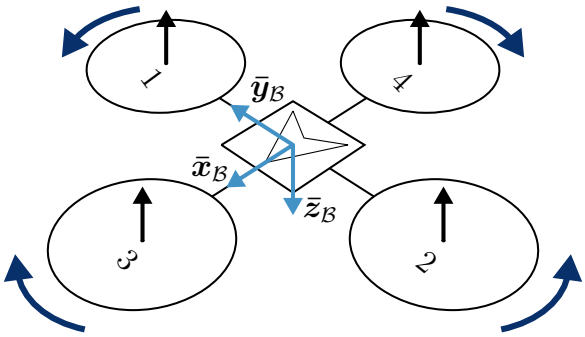


Figure 2.1: Illustration of a quadrotor.

The quadrotor is able to hover in the air by spinning all four motors at the same speed, producing a thrust along the z-axis that overcomes the weight of the vehicle. The quadrotor can gain and lose altitude by adjusting the speed of all four motors equally. The spinning motors cause a torque, and the net torque of the vehicle is negated by spinning motors 1 and 2 in the opposite direction of motors 3 and 4. To move the quadrotor along the y-axis, it needs to perform a roll maneuver by increasing the speed of motor 2 and decreasing the speed of motor 1 with the same amount. Likewise, to move the quadrotor along the x-axis, it needs to perform a pitch maneuver by increasing the speed of motor 4 and decreasing the speed of motor 3 with the same amount. To adjust the heading, the quadrotor needs to perform a yaw maneuver by increasing the speed of motors 3 and 4 and decreasing the speed of motors 1 and 2 by the same amount. These maneuvers are illustrated in Fig. 2.2 [12].

A lot of research has been conducted on the control of multirotors. A survey on different control algorithms for multirotors has recently been done and provides a list of common control schemes along with a summary of each [12]. These control algorithms include linear control techniques

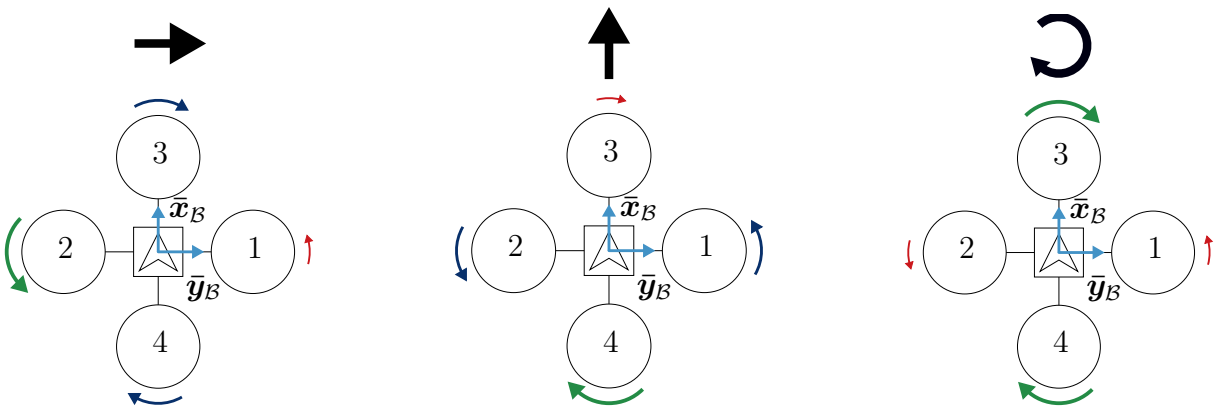


Figure 2.2: A roll, pitch and yaw maneuver.

such as Proportional Integral Derivative (PID) and Linear Quadratic Regulator (LQR), non-linear control techniques such as sliding mode control and feedback linearization, robust and adaptive control techniques and learning-based control techniques such as neural network-based and reinforcement learning-based controllers. The PID controller is the most common because of its simplicity, it is easy to implement and produces good results when tuned correctly. Companies such as DJI and Parrot, as well as open-source flight control software stacks such as PX4, ArduPilot, Betaflight, Cleanflight, and iNav makes use of PID controllers.

2.2 Multirotor with Payload

Many of the applications of multirotors involve some sort of payload. Multirotors are ideal for carrying payloads as they can produce a lot of thrust due to the high number of motors. Hence, a lot of research has been done on using multirotors with payloads. There are mainly two types of payloads: suspended payloads and grasped payloads [13]. A suspended payload is attached to the multirotor using a rope, cable or rigid rod. These payloads are free to swing underneath the multirotor and significantly alters the flight characteristics of the vehicle. A grasped payload is rigidly attached to the multirotor using a gripper, container or a fixed joint, effectively increasing the mass of the vehicle, changing the mass moment of inertia of the system, and possibly changing the center of gravity of the system [14]. A multirotor transporting a grasped payload and suspended payload, respectively, is shown in Fig. 2.3 and Fig. 2.4.

By making use of a grasped payload, the multirotor can be enabled to autonomously pick up the payload for transport with a gripper. However, grasped payloads are limited in size and shape. Suspended payloads, on the other hand, can be any arbitrary size or shape as long as it can be attached to a rope, cable or rod. This enables the multirotor to transport a wider range of payloads compared to using a multirotor for grasped payloads. Different controllers for each of these payloads have been researched and designed.

2.2.1 Grasped Payload

A grasped payload affects the dynamics of the vehicle. It changes the mass moment of inertia of the system, but this is usually negligible and, therefore, the assumption is made in most literature that the payload is a point-mass. The grasped payload mainly affects the vertical axis of the vehicle as weight is added to the multirotor. In general this is not a concern, as

Figure 2.3: Example of a multirotor with a grasped payload [15].

Figure 2.4: Example of a multirotor with a suspended payload [16].

the vertical controllers of the multirotor will compensate for this due to the integral term in a classical PID controller. However, if the integral term of the controller increases significantly, there is a risk of driving the system to instability [17]. Therefore, most of the literature dealing with grasped payloads use an adaptive or robust controller to better handle the parameter uncertainty.

Min et al. [17] proposes an Adaptive Robust Control (ARC) controller to deal with the parameter uncertainty. This controller estimates the mass of the vehicle to use in the feedforward path. In the feedback path, Proportional Derivative (PD) control and a robust control technique are used. It was proven in simulation that this method outperformed classical PID control and could estimate the payload mass fairly well.

Emran et al. [18] approached the problem of a varying payload mass in the use of multirotor add and drop applications. The author designed and implemented a Model Reference Adaptive Control (MRAC) controller that estimates the control parameters to achieve the desired output. The control scheme proved to handle the mass changes in a practical experiment and could track the desired setpoints successfully and with good performance.

2.2.2 Suspended Payload

In the case of a multirotor transporting a suspended payload, there are mainly two types of approaches to minimize the effect of the payload on the multirotor. The first method is to generate swing free trajectories that the multirotor should follow. This method focusses on not inducing any oscillations, caused by the swinging payload, into the system. The second approach is to actively damp these oscillations, as they occur, using an anti-swing controller. This is similar to anti-swing controllers present on cranes [19, 20]. However, in this case the swinging payload greatly affects the multirotor whereas it would not affect the crane as much. Anti-swing controllers have also been designed for traditional helicopters for the transport of payloads [21]. Only recently, methods have been studied for the use of multirotors transporting payloads.

Another consideration for the suspended payload case is the model of the payload. Most literature model the suspended payload as a rigid rod or taut cable attached to the multirotor. This is the simplest model and a valid assumption, as the cable will be taut during most of the flight. There are research groups that try to incorporate the flexibility of the cable into their model and controller. However, these are in the minority as that is a difficult problem and, in most cases, unnecessary.

Minimum Swing Trajectory Generation

Minimum swing trajectory generation addresses the issue that the swing of the payload affects the flight characteristics of the multirotor. By using a trajectory that inherently does not cause large payload oscillations, the flight characteristics of the vehicle will not be altered as much.

One approach to achieve this is by using dynamic programming. Palunko et al. [22] made use of this technique to generate optimal trajectories for swing-free maneuvers. Dynamic programming is an open-loop optimization method, where a series of decisions are generated to produce the optimal case. The authors applied this technique in both simulation and a practical setup and achieved good results as it reduced the swing angles.

Another approach to generate minimum swing trajectories is by using the input shaping technique. Sadr et al. [13] made use of this technique, which involves the convolution of the desired waypoints and a set of impulses. The goal of the impulses is to reduce the swing angle of the payload. The initial impulse induces an angle and the second impulse is applied at exactly the right time to negate the induced angle, as shown in Fig. 2.5. This is similar to the concept of destructive interference. The authors were able to improve the path tracking performance, in simulation, of the multirotor and suspended payload, with this method.

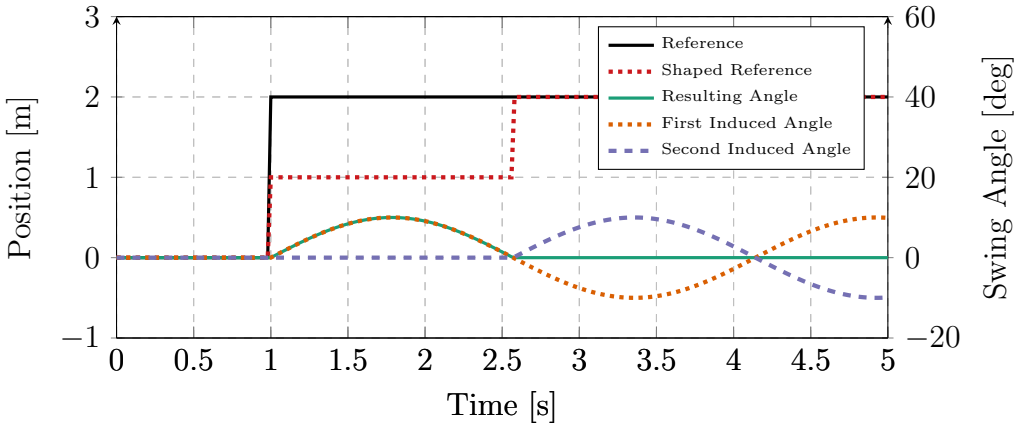


Figure 2.5: Input shaping with deconstructive superposition.

The abovementioned methods require a very accurate model of the system as they are open-loop techniques. These methods are not robust, as they suffer from model uncertainties and external disturbances.

Wang and Xian [23], therefore, proposed an online trajectory generation technique that makes use of feedback. The proposed algorithm contains two parts: a positioning target trajectory component and an anti-swing component. The anti-swing component produces a trajectory based on the current state of the payload. The authors achieved accurate position tracking of the multirotor in simulation and it proved to be more robust. This technique does, however, require real-time knowledge of the payload's state.

Anti-Swing Controllers

Anti-swing controllers are designed to actively damp the oscillations caused by the payload. These methods are all based on feedback control. Various feedback control techniques have

been researched and designed to solve this problem.

Intuitively, one wants to keep the payload angles small and eventually control them to zero. Alothman et al. [24] designed an LQR controller to achieve this goal. They modeled the cable with a transitioning function to transition from slack to taut. The LQR controller is designed around hover and applied in a simulation environment where the vehicle takes off and the cable transitions from slack to taut. The simulation results show an improvement of the LQR controller compared to a traditional PD controller.

Goodarzi et al. [25] proposes a geometric control technique for the multirotor and payload system. The authors modeled the payload as a system of serially-connected links to represent a cable. A coordinate-free form of the equations of motion of the system was derived, meaning that the complete equations of motion are represented in the inertial frame, to minimize the complexity of the model. Thus, there are no singularities that can occur with coordinate frame transformations. The control algorithm focusses on stabilizing the position of the vehicle while aligning the payload links vertically below the vehicle. They designed a non-linear controller that achieved this in a practical setup.

The abovementioned methods focus on keeping the payload vertically below the multirotor with small swing angles. However, the problem is twofold: one wants the multirotor to track a path while maintaining small payload swing angles. Therefore, some research groups have proposed hybrid controllers with one part handling the multirotor position and another handling the payload angles. A simple block diagram illustrating the architecture of such a hybrid controller is shown in Fig. 2.6.

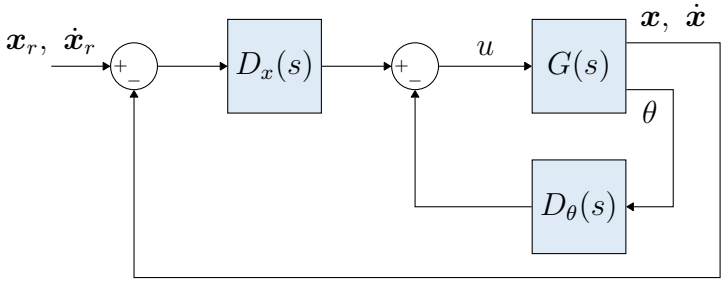


Figure 2.6: Block diagram of hybrid control system for multirotor with suspended payload.

Yang and Xian [26] used such a technique by designing separate controllers, one for the positioning of the vehicle and one for the swing angle of the payload. The authors reduced the problem to the 2-Dimensional (2D) case, unlike the other 3-Dimensional (3D) solutions, and used backstepping control for the positioning of the vehicle and a Robust Integral of the Sign of the Error (RISE) based controller for the swing angle of the payload. The algorithm was tested in simulation and the multirotor was able to track the position with minimal disturbances.

Raffo and De Almeida [27] made use of H-Infinity ($\mathscr{H}_\infty$) control for the positioning of the multirotor and through Lyapunov redesign added a component to reduce the swing of the payload. They modeled the system as a pendulum attached to the multirotor. The $\mathscr{H}_\infty$ controller was chosen for the positioning of the vehicle because of its robustness. It was found in simulation that the vehicle was able to track the given path, however, the payload still experienced a lot of swing. Therefore, another control law was designed through Lyapunov redesign and summed to the $\mathscr{H}_\infty$ controller. This showed an improvement in the swing angles of the payload.

2.3 Multirotor with Unknown Payload

In multirotor transportation applications, the vehicle should transport a variety of different payloads. Whether the application is consumer delivery or medical emergency needs, the payload will be different with each flight. Therefore, the vehicle needs to be able to adapt to each payload transported.

This is similar to the solutions proposed in [17] and [18] with the grasped payload case. However, suspended payloads are easier to attach and detach from the multirotor and is more versatile with respect to the shape and size of the payload that the vehicle can carry. Nonetheless, suspended payloads are much harder to control and subsequently receives a lot of attention in literature.

Consider the multirotor and payload in Fig. 2.7. The payload's parameters refer to its physical properties, like the payload mass, m, and the cable length, l. The payload's states refer to its position and/or the swing angles, θ and ϕ, of the payload.

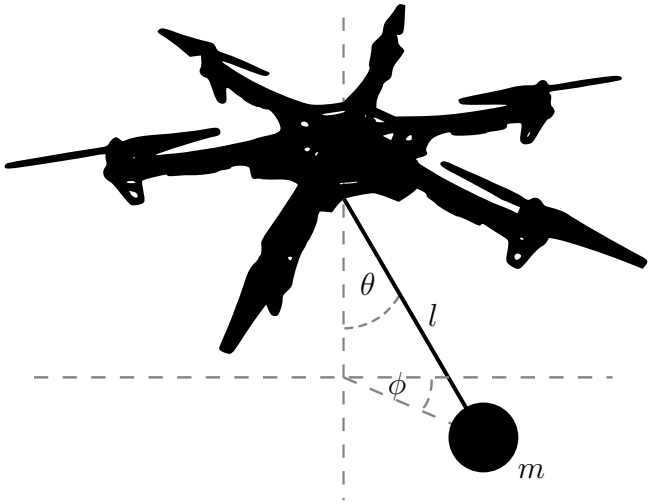


Figure 2.7: Multirotor with a suspended payload.

When considering an unknown suspended payload, one should take into account the sensors that are available on the vehicle. One can use sensors to estimate the position of the swinging payload and extract the suspended payload's parameters. Another option is to not add any extra sensors and estimate the payload's state, given knowledge of the payload's parameters. Estimating both the payload's state and parameters simultaneously is a difficult task.

2.3.1 Unknown Payload Parameters

During the controller design of a quadrotor and payload system, the parameters of the payload are needed. Therefore, if these parameters can be estimated, the controller can be adapted to the specific payload.

Dai et al. [28] extended the work in [25] to include an unknown payload mass. The length of the cable is still known, but a Retrospective Cost Adaptive Control (RCAC) algorithm is implemented to compensate for the unknown payload mass.

Yang and Xian [29], on the other hand, focused on the problem with an unknown cable length, but a known payload mass. The authors also implemented an adaptive controller to compensate for the unknown cable length and showed good results in a practical experiment.

It is quite difficult to estimate the payload's parameters in a practical setup as one needs information about the payload's state. In most literature, this is done by using a motion capture system [25, 29]. This involves the use of offboard cameras to determine the pose of the multirotor and the payload. However, this cannot be done out in the field and, thus, is only applicable to certain use-cases.

2.3.2 Unknown Payload States

The swing angles are crucial bits of information as it gives an indication of the magnitude of the disturbance force, from the payload, acting on the vehicle. One would like to design a controller to keep these angles as low as possible.

To design a controller like this, one needs estimates of the payload swing angles. Bisgaard et al. [21] used a downward-facing camera on a helicopter to estimate the swing angles and cable length of the payload. The authors continued to use these values in an input shaper to avoid oscillations and a feedback controller to further dampen the oscillations. This showed promising results in a practical experiment. However, the input shaper considerably reduced the response time of the system.

De Angelis [30] investigated an estimator to achieve this task with no extra sensors attached to the vehicle. The author used the sensors already present on the multirotor and knowledge of the payload's parameters to estimate the state of the payload. The author successfully demonstrated the estimator and fed back the estimated angles to control the multirotor and suspended payload system.

Palunko et al. [14] approached the problem a bit differently. The authors designed an adaptive controller, given the payload's parameters, to estimate the center of gravity of the system and adapt the controller accordingly. They present simulation results showcasing the center of gravity estimates and the reduced payload swing angles. The authors designed the controller in this way such that it can handle both suspended payloads and grasped payloads that might change the center of gravity of the system.

All of the abovementioned literature estimates the states of the payload and applies a control input accordingly. Guerrero-Sánchez et al. [31] designed a control law that does not depend on the swing angles of the payload. The control law makes use of Interconnection and Damping Assignment-Passivity Based Control. The authors presented results from a practical experiment showcasing the reduced payload angles. The algorithm requires knowledge of the payload's parameters, but is quite robust against parameter uncertainty.

2.4 Summary

This chapter explored a wide range of solutions in literature, aimed at transporting a payload using a multirotor. These solutions include different types of payloads and different control algorithms. A diagram showcasing these solutions and their intended application is illustrated in Fig. 2.8.

The trends noticed during this literature study are:

- Adaptive control is often used to simultaneously estimate and control an unknown system.
- Estimators are built to estimate unknown parameters or unknown states, which are then used in the control system.

These trends serve as a starting point to investigate possible solutions for the stabilization of a multirotor with an unknown suspended payload.

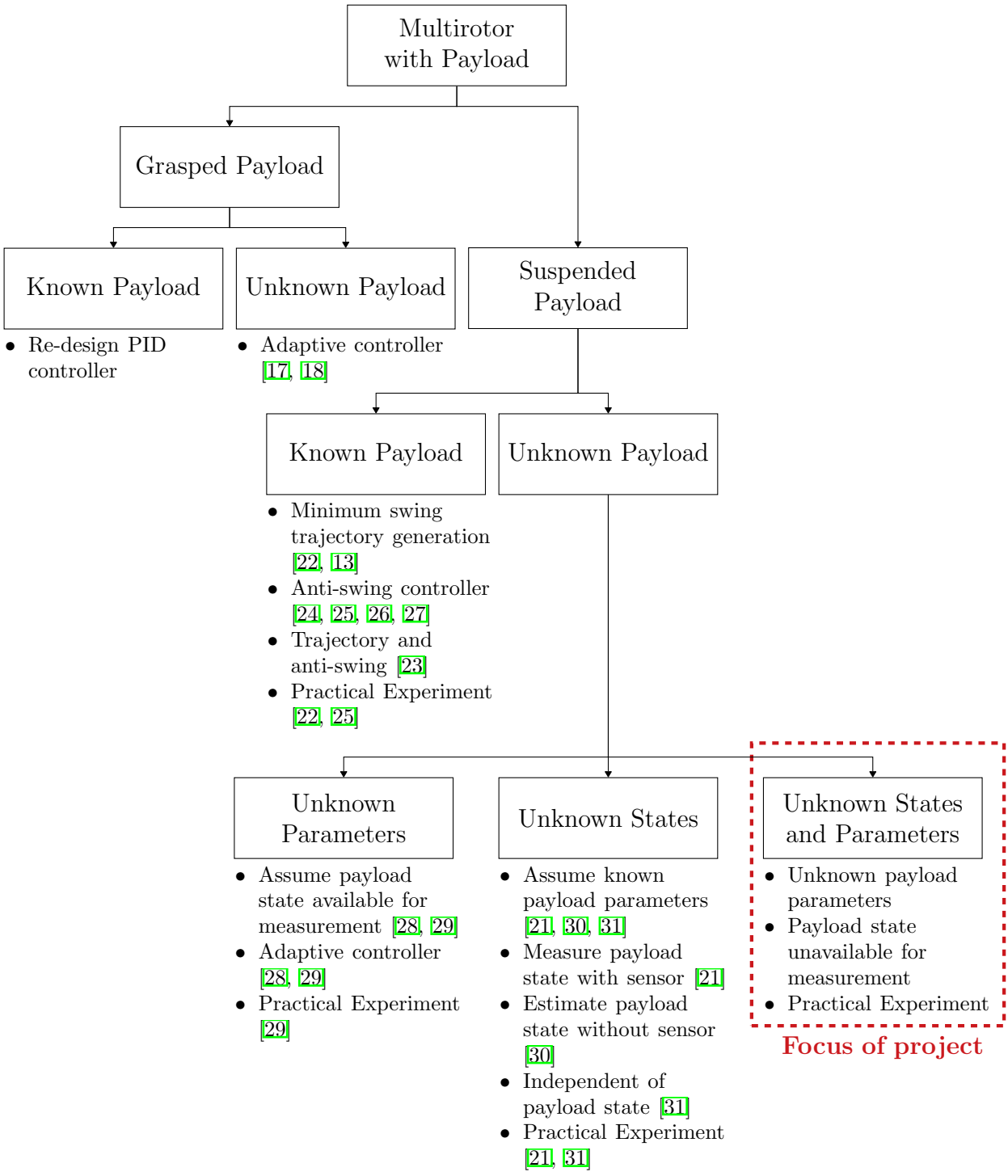


Figure 2.8: Summary of literature study on multirotors with payloads

3. Modelling

In this chapter, the non-linear model of a RUAV with a suspended payload will be derived. The derived model is used in the following sections to simulate the vehicle and design the controllers.

The RUAV used in this project is a quadrotor. More specifically, the quadrotor is in the X configuration as shown in Fig. 3.1. In an X configurated quadrotor, the front of the vehicle is situated between two of the propellers.

This chapter first defines the different coordinate frames. An overview of quaternions will be given thereafter, as the attitude of the vehicle is represented by quaternions. Lastly, the six degrees of freedom equations of motion and the different forces and moments acting in on the vehicle will be described.

3.1 Coordinate Frames

Consider the quadrotor shown in Fig. 3.1. Two coordinate frames are of interest, the inertial or earth coordinate frame, indicated by $\mathcal{I} = \{\bar{\boldsymbol{x}}_{\mathcal{I}}, \bar{\boldsymbol{y}}_{\mathcal{I}}, \bar{\boldsymbol{z}}_{\mathcal{I}}\}$, and the body coordinate frame, indicated by $\mathcal{B} = \{\bar{\boldsymbol{x}}_{\mathcal{B}}, \bar{\boldsymbol{y}}_{\mathcal{B}}, \bar{\boldsymbol{z}}_{\mathcal{B}}\}$.

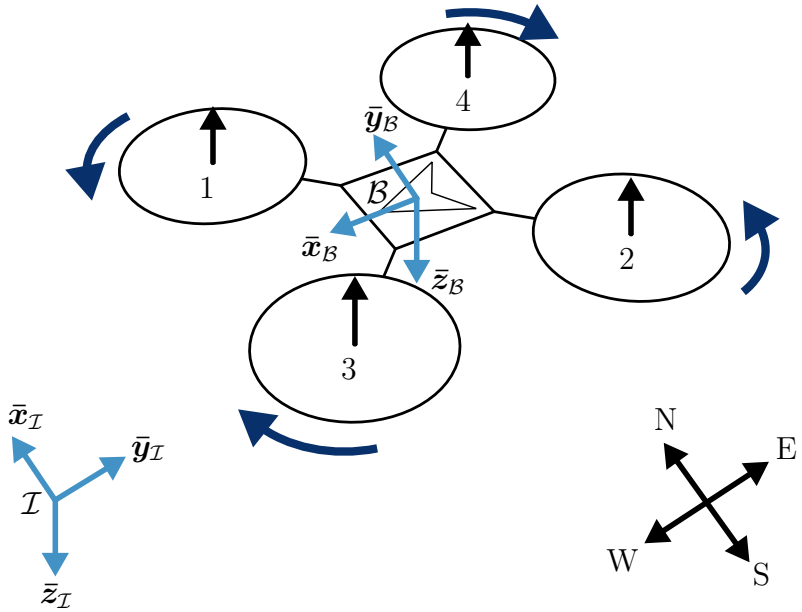


Figure 3.1: The inertial and body frames.

A standard North-East-Down (NED) axis system is used for the inertial coordinate frame. This assumes a flat non-rotating earth, which is a valid assumption as the quadrotor will not travel such large distances where the curvature of the earth needs to be taken into account. The origin of the inertial frame is chosen as the takeoff location of the quadrotor. The x-axis is pointing in the North direction, the y-axis in the East direction and the z-axis in the Down direction.

The body coordinate frame is fixed to the quadrotor with the origin at the CoM of the vehicle. The x-axis is pointing to the front of the vehicle, the y-axis to the right and the z-axis in the downwards direction. The body frame is defined in the inertial frame as a displacement from the origin with a rotation.

In the next section quaternions will be discussed, which provides a way to describe the rotation between the body and inertial frames.

3.2 Quaternions

Quaternions, also known as Euler Parameters, is a set of non-singular attitude coordinates [32]. Consider the rotation about the unit vector $\bar{\boldsymbol{r}} = [r_x \quad r_y \quad r_z]^T$ by θ, as shown in Fig. 3.2.

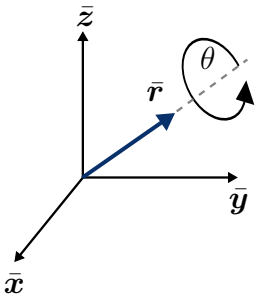


Figure 3.2: Axis Angle Rotation.

The quaternion, $\bar{\boldsymbol{q}} = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$, is defined from this rotation as

$$q_0 = \cos\left(\tfrac{\theta}{2}\right), \tag{3.1}$$
$$q_1 = r_x \sin\left(\tfrac{\theta}{2}\right), \tag{3.2}$$
$$q_2 = r_y \sin\left(\tfrac{\theta}{2}\right), \text{ and} \tag{3.3}$$
$$q_3 = r_z \sin\left(\tfrac{\theta}{2}\right). \tag{3.4}$$

The quaternion satisfies

$$|\bar{\boldsymbol{q}}| = \sqrt{q_0^2 + q_1^2 + q_3^2 + q_4^2} = 1, \tag{3.5}$$

which is known as a unit quaternion.

Quaternions consist of two parts, namely a magnitude, q_0, and a vector, $\boldsymbol{q}_v = [q_1 \quad q_2 \quad q_3]^T$. The quaternion representation in this thesis defines q_0 as the magnitude part of the quaternion.

Quaternions are difficult to visualize as the vector does not give an intuition of the orientation. Other attitude coordinates, such as 3-2-1 Euler Angles, are more intuitive and easy to visualize, but suffers from a singularity, in this case at a $\pm 90°$ pitch angle. Quaternions do not have any mathematical singularities.

However, the representation of a quaternion is not unique, as there are two quaternions describing the same rotation. Consider a rotation of θ about $\bar{\boldsymbol{r}}$. The same rotation can be described with an angle of $\phi = \theta - 2\pi$. One rotation corresponds to the shorter rotation and the other is the same as rotating the long way around, both resulting in the same end position. The quaternions describing this rotation is given by Eq. (3.6).

$$\bar{\boldsymbol{q}}_1(\bar{\boldsymbol{r}}, \theta) = \begin{bmatrix} q_0 \\ \boldsymbol{q}_v \end{bmatrix} \qquad \bar{\boldsymbol{q}}_2(\bar{\boldsymbol{r}}, \phi) = \begin{bmatrix} \cos\left(\frac{\theta}{2} - \pi\right) \\ \bar{\boldsymbol{r}} \sin\left(\frac{\theta}{2} - \pi\right) \end{bmatrix} = \begin{bmatrix} -\cos\left(\frac{\theta}{2}\right) \\ -\bar{\boldsymbol{r}} \sin\left(\frac{\theta}{2}\right) \end{bmatrix} = \begin{bmatrix} -q_0 \\ -\boldsymbol{q}_v \end{bmatrix} \tag{3.6}$$

The quaternions $\bar{\boldsymbol{q}}_1$ and $\bar{\boldsymbol{q}}_2$ describe the same rotation. In this case, the vehicle will never perform a long rotation and therefore the shortest rotation representation will always be used.

3.2.1 Inverse Quaternion

A quaternion describes a rotation from one coordinate frame to another, and the inverse of a quaternion describes the inverse rotation. The inverse of a quaternion is given as

$$\bar{\boldsymbol{q}}^{-1} = \bar{\boldsymbol{q}}(\bar{\boldsymbol{r}}, -\theta) = \begin{bmatrix} q_0 \\ -\boldsymbol{q}_v \end{bmatrix}. \tag{3.7}$$

3.2.2 Quaternion Multiplication

Two consecutive rotations are described by multiplying the quaternion sets. The multiplication

$$\bar{\boldsymbol{q}}' = \bar{\boldsymbol{q}}_b \bar{\boldsymbol{q}}_a \tag{3.8}$$

describes a rotation consisting of firstly rotating by $\bar{\boldsymbol{q}}_a$ and then by $\bar{\boldsymbol{q}}_b$. The formula describing the multiplication of quaternions is given as

$$\bar{\boldsymbol{q}}' = \bar{\boldsymbol{q}}_b \bar{\boldsymbol{q}}_a = \begin{bmatrix} q_{a,0}q_{b,0} - q_{a,1}q_{b,1} - q_{a,2}q_{b,2} - q_{a,3}q_{b,3} \\ q_{a,0}q_{b,1} + q_{a,1}q_{b,0} - q_{a,2}q_{b,3} + q_{a,3}q_{b,2} \\ q_{a,0}q_{b,2} + q_{a,1}q_{b,3} + q_{a,2}q_{b,0} - q_{a,3}q_{b,1} \\ q_{a,0}q_{b,3} - q_{a,1}q_{b,2} + q_{a,2}q_{b,1} + q_{a,3}q_{b,0} \end{bmatrix}. \tag{3.9}$$

3.3 Six Degrees of Freedom

The quadrotor vehicle has six degrees of freedom as it is free to move in all three directions of the axis system and it is also free to rotate about any of the three axes. The equations of motion describing a six degrees of freedom model is fully derived in [33] using Euler angles, instead of quaternions. In this section, the model will be introduced and discussed.

3.3.1 Kinematics

Kinematics refer to the motion of a vehicle, which is described by its acceleration, velocity, and position. The inertial linear velocity and position are of interest. The linear velocity can be obtained by transforming the body linear velocity, defined in Eqs. (3.21) - (3.23), into the inertial frame. The attitude of the vehicle is required to perform such a transformation, which in this case is defined with quaternions. It is obtained from the body angular rates, defined in Eqs. (3.24) - (3.26), by using

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \Omega_{\mathcal{B}_X} \\ \Omega_{\mathcal{B}_Y} \\ \Omega_{\mathcal{B}_Z} \end{bmatrix}. \tag{3.10}$$

The full derivation of Eq. (3.10) can be found in [32]. The attitude can now be used to transform the body linear velocity to the inertial linear velocity with

$$\boldsymbol{V}_{\mathcal{I}} = \boldsymbol{R}_V^{-1} \boldsymbol{V}_{\mathcal{B}} \tag{3.11}$$

where

$$\boldsymbol{R}_V = \begin{bmatrix} q_0^2+q_1^2+q_2^2+q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}. \tag{3.12}$$

$\boldsymbol{R}_V$ is the transformation matrix, or otherwise known as the Direct Cosine Matrix (DCM), describing a rotation from the body frame to the inertial frame. The derivation of the DCM from quaternions can be found in [32]. The inertial position of the vehicle can now be obtained by integrating the inertial linear velocity.

3.3.2 Kinetics

Kinetics refer to the relation of forces and moments acting on an object and its acceleration, velocity, and position. Using Newton's second law of motion, the equations

$$\boldsymbol{F}_{\mathcal{B}} = m_q\dot{\boldsymbol{V}}_{\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}} \times m_q\boldsymbol{V}_{\mathcal{B}} \text{ and} \tag{3.13}$$

$$\boldsymbol{M}_{\mathcal{B}} = \boldsymbol{I}_q\dot{\boldsymbol{\Omega}}_{\mathcal{B}} + \boldsymbol{\Omega}_{\mathcal{B}} \times \boldsymbol{I}_q\boldsymbol{\Omega}_{\mathcal{B}} \tag{3.14}$$

are obtained, where,

$$\boldsymbol{F}_{\mathcal{B}} = [F_{\mathcal{B}_X} \quad F_{\mathcal{B}_Y} \quad F_{\mathcal{B}_Z}]^T \text{ and} \tag{3.15}$$

$$\boldsymbol{M}_{\mathcal{B}} = [M_{\mathcal{B}_X} \quad M_{\mathcal{B}_Y} \quad M_{\mathcal{B}_Z}]^T \tag{3.16}$$

are the forces and moments, respectively, acting on the body frame of the quadrotor. The derivatives in the above equations are all with respect to the body frame of the quadrotor. The total mass of quadrotor is given by m_q and

$$\boldsymbol{I}_q = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.17}$$

is the mass moment of inertia of the quadrotor. The quadrotor is assumed to be symmetrical about the XZ- and YZ-plane. Therefore, the mass moment of inertia of the vehicle simplifies to

$$\boldsymbol{I}_q \approx \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{3.18}$$

The quadrotor's linear velocity and angular velocity is given by

$$\boldsymbol{V}_{\mathcal{B}} = [V_{\mathcal{B}_X} \quad V_{\mathcal{B}_Y} \quad V_{\mathcal{B}_Z}]^T \text{ and} \tag{3.19}$$

$$\boldsymbol{\Omega}_{\mathcal{B}} = [\Omega_{\mathcal{B}_X} \quad \Omega_{\mathcal{B}_Y} \quad \Omega_{\mathcal{B}_Z}]^T. \tag{3.20}$$

Simplifying the above equations yields the set of differential equations (3.21) - (3.26) describing the kinetics of the quadrotor vehicle.

$$\dot{V}_{\mathcal{B}_X} = \frac{F_{\mathcal{B}_X}}{m_q} - V_{\mathcal{B}_Z}\Omega_{\mathcal{B}_Y} + V_{\mathcal{B}_Y}\Omega_{\mathcal{B}_Z} \tag{3.21}$$

$$\dot{V}_{\mathcal{B}_Y} = \frac{F_{\mathcal{B}_Y}}{m_q} - V_{\mathcal{B}_X}\Omega_{\mathcal{B}_Z} + V_{\mathcal{B}_Z}\Omega_{\mathcal{B}_X} \tag{3.22}$$

$$\dot{V}_{\mathcal{B}_Z} = \frac{F_{\mathcal{B}_Z}}{m_q} - V_{\mathcal{B}_Y}\Omega_{\mathcal{B}_X} + V_{\mathcal{B}_X}\Omega_{\mathcal{B}_Y} \tag{3.23}$$

$$\dot{\Omega}_{\mathcal{B}_X} = \frac{M_{\mathcal{B}_X} + \Omega_{\mathcal{B}_Y}\Omega_{\mathcal{B}_Z}\left(I_{yy} - I_{zz}\right)}{I_{xx}} \tag{3.24}$$

$$\dot{\Omega}_{\mathcal{B}_Y} = \frac{M_{\mathcal{B}_Y} + \Omega_{\mathcal{B}_X}\Omega_{\mathcal{B}_Z}\left(I_{zz} - I_{xx}\right)}{I_{yy}} \tag{3.25}$$

$$\dot{\Omega}_{\mathcal{B}_Z} = \frac{M_{\mathcal{B}_Z} + \Omega_{\mathcal{B}_X}\Omega_{\mathcal{B}_Y}\left(I_{xx} - I_{yy}\right)}{I_{zz}} \tag{3.26}$$

These kinetic and kinematic equations fully describe the position and attitude of the vehicle given the forces and moments that act on it. Next, these forces and moments will be introduced and described.

3.4 Forces and Moments

The different forces and moments acting in on the vehicle are the actuators, gravity, aerodynamics and the payload. The total forces and moments are thus given by

$$\boldsymbol{F}_{\mathcal{B}} = \boldsymbol{F}_{\mathcal{B}}^{\mathrm{T}} + \boldsymbol{F}_{\mathcal{B}}^{\mathrm{G}} + \boldsymbol{F}_{\mathcal{B}}^{\mathrm{A}} + \boldsymbol{F}_{\mathcal{B}}^{\mathrm{P}} \text{ and} \tag{3.27}$$

$$\boldsymbol{M}_{\mathcal{B}} = \boldsymbol{M}_{\mathcal{B}}^{\mathrm{T}} + \boldsymbol{M}_{\mathcal{B}}^{\mathrm{G}} + \boldsymbol{M}_{\mathcal{B}}^{\mathrm{A}} + \boldsymbol{M}_{\mathcal{B}}^{\mathrm{P}}, \tag{3.28}$$

where the superscripts T, G, A and P denote the actuators producing thrust, gravity, aerodynamics and the effect of the suspended payload on the vehicle, respectively.

3.4.1 Actuators

The actuators of the vehicle are the four motor-propeller pairs on the vehicle. Each of the motor-propeller pairs produces a thrust, T_i where $i = \{1, 2, 3, 4\}$ denotes the motor number, as indicated by Fig. 3.1, depending on the rotational speed of the motor. The thrust produced by these pairs is modelled as a first order differential equation given by

$$\dot{T}_i = \frac{-T_i + T_{i_R}}{\tau}, \tag{3.29}$$

where T_{i_R} is the reference thrust and τ is the time constant of the motor-propeller pair. The forces and moments acting on the vehicle due to the actuators are given by the equations

$$F_{\mathcal{B}_Z}^{\mathrm{A}} = T_1 + T_2 + T_3 + T_4, \tag{3.30}$$

$$M_{\mathcal{B}_X}^{\mathrm{A}} = \frac{d}{\sqrt{2}}\left(-T_1 + T_2 + T_3 - T_4\right), \tag{3.31}$$

$$M_{\mathcal{B}_Y}^{\mathrm{A}} = \frac{d}{\sqrt{2}}\left(T_1 - T_2 + T_3 - T_4\right), \text{ and} \tag{3.32}$$

$$M_{\mathcal{B}_Z}^{\mathrm{A}} = R_N\left(T_1 + T_2 - T_3 - T_4\right), \tag{3.33}$$

where d is the distance from the center of the vehicle to a motor, and R_N is the virtual yaw moment arm. The virtual yaw moment arm relates the thrust produced by a motor to a torque. Virtual actuators are defined because of the high coupling of the actuators and the axes of motion. These are referred to as the throttle, aileron, elevator, and rudder, respectively, as defined for a fixed-wing UAV. The virtual actuators relate to the actual actuators by

$$\delta_T = T_1 + T_2 + T_3 + T_4, \tag{3.34}$$

$$\delta_A = \frac{1}{\sqrt{2}}\left(-T_1 + T_2 + T_3 - T_4\right), \tag{3.35}$$

$$\delta_E = \frac{1}{\sqrt{2}}\left(T_1 - T_2 + T_3 - T_4\right), \text{ and} \tag{3.36}$$

$$\delta_R = T_1 + T_2 - T_3 - T_4. \tag{3.37}$$

A matrix, known as the mixing matrix, is defined to transform the actuators to the virtual actuators and vice versa. It is given by

$$\begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 1 & 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}, \text{ with} \tag{3.38}$$

$$\boldsymbol{\delta}_V = \boldsymbol{K}_M \boldsymbol{T}_A \text{ and} \tag{3.39}$$

$$\boldsymbol{T}_A = \boldsymbol{K}_M^{-1} \boldsymbol{\delta}_V. \tag{3.40}$$

The resulting forces and moments equations due to the actuators are

$$\boldsymbol{F}_{\mathcal{B}}^{\mathrm{T}} = \delta_T \bar{\boldsymbol{z}}_{\mathcal{B}} \text{ and} \tag{3.41}$$

$$\boldsymbol{M}_{\mathcal{B}}^{\mathrm{T}} = d\delta_A \bar{\boldsymbol{x}}_{\mathcal{B}} + d\delta_E \bar{\boldsymbol{y}}_{\mathcal{B}} + R_N \delta_R \bar{\boldsymbol{z}}_{\mathcal{B}}. \tag{3.42}$$

3.4.2 Gravity

Gravity is a force acting on the vehicle in the vertical axis of the inertial frame. The magnitude and direction of the force are completely known in the inertial frame. However, the forces in Equations 3.21 - 3.23 are in the body frame. Therefore, the DCM is used to transform the force into the body frame. The total forces and moments acting on the vehicle due to gravity are

$$\boldsymbol{F}_{\mathcal{B}}^{\mathrm{G}} = \boldsymbol{R}_V \begin{bmatrix} 0 \\ 0 \\ m_q g \end{bmatrix} \text{ and} \tag{3.43}$$

$$\boldsymbol{M}_{\mathcal{B}}^{\mathrm{G}} = \boldsymbol{0}. \tag{3.44}$$

3.4.3 Aerodynamics

The aerodynamic forces and moments acting on the vehicle are the most difficult to model accurately and to obtain the parameters from a physical vehicle. The aerodynamic drag forces that the vehicle experience is that of the relative motion of the quadrotor body and air. As the vehicle flies faster, it will experience a larger drag force in the opposite direction of motion. The aerodynamic model is based on previous work done by [34]. The aerodynamic forces are modeled using fluid mechanics, where an object experiences pressure when moving through a fluid. This is described by

$$\boldsymbol{F}_{\boldsymbol{\mathcal{B}}}^{\mathrm{A}} = \frac{1}{2}\rho \boldsymbol{V}_{\mathcal{B}\mathcal{W}} \left|\boldsymbol{V}_{\mathcal{B}\mathcal{W}}\right| \begin{bmatrix} C_{D_X} \\ C_{D_Y} \\ C_{D_Z} \end{bmatrix}, \tag{3.45}$$

where ρ is the air density, $\boldsymbol{V}_{\mathcal{B}\mathcal{W}}$ is the relative velocity of the body frame of the vehicle and wind and C_{D_X}, C_{D_Y} and C_{D_Z} are the lumped drag coefficients and reference area of the vehicle

in the specified axis. The air density chosen is $\rho = 1.225$ kg/m^3, which is the air density at mean sea level and 15°C. The relative velocity, $\boldsymbol{V}_{\mathcal{BW}}$, is calculated as

$$\boldsymbol{V}_{\mathcal{BW}} = -\boldsymbol{V}_{\mathcal{B}} + \boldsymbol{R}_V \boldsymbol{V}_{\mathcal{W}}, \tag{3.46}$$

where $\boldsymbol{V}_{\mathcal{W}}$ is the velocity of the instantaneous wind in the inertial frame.

3.4.4 Suspended Payload

The considered suspended payload is only allowed to swing in one axis. Therefore, the effect of the payload on the vehicle is reduced to the 2D quadrotor and payload problem. The approach followed to determine the forces and moments that the payload exerts on the vehicle, is to solve the 2D problem and augment the solution to the 3D quadrotor model.

Consider the quadrotor and payload model shown in Fig. 3.3, where m_q is the mass of the quadrotor, m_p is the mass of the payload, l is the length of the payload link, k and c are spring and damper coefficients, respectively, $f_{\mathcal{W}}$ is the aerodynamic drag force experienced by the payload, and β is the swing angle. The following assumptions are made regarding the 2D payload:

- The payload link is rigid and weightless.
- The payload is considered to be a point mass.
- The payload is attached to the CoM of the vehicle.
- The payload angle is restricted to $-\frac{\pi}{2} \leq \beta \leq \frac{\pi}{2}$.
- The payload does not experience any aerodynamic drag in the vertical axis.

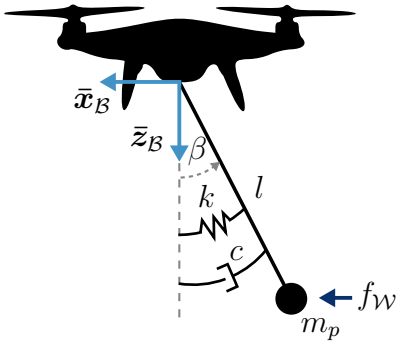


Figure 3.3: 2D Quadrotor and payload model.

It is clear from the last assumption that only the linear movement of the vehicle will affect the payload and the vehicle's rotation will have no effect. Therefore, the quadrotor considered in this 2D model is restricted to only move in the horizontal and vertical directions. It is assumed that the rotational controllers will stabilize the quadrotor's attitude instantaneously, as the rotational dynamics are faster than the translational dynamics of the vehicle. Therefore, the rotation of the vehicle is negated in the reduced 2D model.

Lagrangian mechanics is used to derive the mathematical model of the 2D quadrotor and payload model. A brief description of Lagrangian mechanics is given next, followed by the mathematical derivation of the model.

Lagrangian Mechanics

Lagrangian mechanics is an energy-based approach to obtain the differential equations describing a system. A quantity, called the Lagrangian, is defined as

$$\mathcal{L} = T_e - V_e, \tag{3.47}$$

where T_e is the total kinetic energy of the system and V_e is the total potential energy of the system [35]. The Euler-Lagrange equation, given as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{p}_i}\right) - \frac{\partial \mathcal{L}}{\partial p_i} = Q_i, \tag{3.48}$$

fully describes the system consisting of the generalized coordinates

$$\boldsymbol{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}, \tag{3.49}$$

where $i = \{1, 2, \ldots, n\}$ and Q_i is the non-conservative force of the i_{th} generalized coordinate.

The Euler-Lagrange equation yields a set of coupled differential equations describing the system. The equations of motion describing a simple pendulum is derived to illustrate the use of Lagrangian Mechanics. Thereafter, the problem is extended to the quadrotor and suspended payload system, of which the equations of motion are also derived using Lagrangian Mechanics.

Simple Pendulum

Consider the pendulum model shown in Fig. 3.4, where l is the length of the pendulum, m is the point mass at the end of the pendulum and β is the angle of the pendulum. The total kinetic energy of the system is calculated by

$$T_e = \frac{1}{2} m \left(\dot{x}^2 + \dot{z}^2\right), \tag{3.50}$$

where x and z denotes the horizontal and vertical position of the point mass, respectively. The positions and velocites of the point mass is given as

$$\begin{aligned} x &= -l\sin\beta, \\ \dot{x} &= -l\dot{\beta}\cos\beta, \\ z &= l\cos\beta, \text{ and} \\ \dot{z} &= -l\dot{\beta}\sin\beta. \end{aligned} \tag{3.51}$$

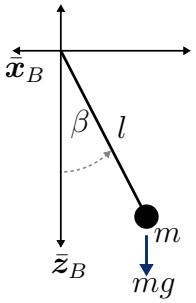


Figure 3.4: Pendulum model.

Substituting Eq. (3.51) into Eq. (3.50) yields the total kinetic energy of the system, given as

$$\begin{aligned} T_e &= \frac{1}{2}m\left[\left(-l\dot{\beta}\cos\beta\right)^2 + \left(-l\dot{\beta}\sin\beta\right)^2\right] \\ &= \frac{1}{2}ml^2\dot{\beta}^2. \end{aligned} \tag{3.52}$$

The total potential energy of the system is calculated as

$$V_e = -mgl\cos\beta, \tag{3.53}$$

yielding the Lagrangian,

$$\mathcal{L} = T_e - V_e = \frac{1}{2}ml^2\dot{\beta}^2 + mgl\cos\beta. \tag{3.54}$$

In this case only one generalized coordinate is considered, yielding $\boldsymbol{p} = [\beta]$. The Euler-Lagrange equation then becomes

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\beta}}\right) - \frac{\partial\mathcal{L}}{\partial\beta} = 0, \tag{3.55}$$

with no non-conservative forces working in on the system. The different terms of the Euler-Lagrange equation is calculated as

$$\begin{aligned} &\frac{\partial\mathcal{L}}{\partial\dot{\beta}} = ml^2\dot{\beta}, \\ &\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\beta}}\right) = ml^2\ddot{\beta}, \text{ and} \\ &\frac{\partial\mathcal{L}}{\partial\beta} = -mgl\sin\beta. \end{aligned} \tag{3.56}$$

Substituting Eq. (3.56) into Eq. (3.55) yields

$$\ddot{\beta} + \frac{g}{l}\sin\beta = 0, \tag{3.57}$$

which is the equation for a simple pendulum model and the same result can be obtained by using Newton's second law of motion. This simple example illustrates the derivation of the equations of motion of a system using Lagrangian mechanics. The quadrotor and payload system is only an extension of the simple pendulum model. The pendulum is no longer fixed at the origin, but is rather fixed to a moving body, the quadrotor, free to move in both the horizontal and vertical directions. The pendulum is also extended by the addition of a spring and damper system. The derivation of the quadrotor and payload system using Lagrangian mechanics is described in the next section.

2D Quadrotor with Suspended Payload Equations of Motion

The set of generalized coordinates under consideration for the quadrotor and payload system, shown in Fig. 3.3, is

$$\boldsymbol{p} = \begin{bmatrix} x_q \\ z_q \\ \beta \end{bmatrix}, \tag{3.58}$$

where x_q and z_q are the horizontal and vertical positions of the quadrotor, respectively, and β is the swing angle of the payload. The total kinetic energy of the system is calculated by

$$T_e = \frac{1}{2}m_q\left(\dot{x_q}^2 + \dot{z_q}^2\right) + \frac{1}{2}m_p\left(\dot{x_p}^2 + \dot{z_p}^2\right), \tag{3.59}$$

where x_p and z_p are the horizontal and vertical positions of the payload, respectively. The positions and velocities of the payload are calculated by

$$x_p = x_q - l\sin\beta, \tag{3.60}$$
$$\dot{x_p} = \dot{x_q} - l\dot{\beta}\cos\beta, \tag{3.61}$$
$$z_p = z_q + l\cos\beta, \text{ and} \tag{3.62}$$
$$\dot{z_p} = \dot{z_q} - l\dot{\beta}\sin\beta. \tag{3.63}$$

The total potential energy of the system is

$$V_e = -m_p g z_p. \tag{3.64}$$

The non-conservative forces of the system are the effects of the spring and damper. The set of non-conservative forces are described by

$$\boldsymbol{Q} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -k\beta - c\dot{\beta} - l f_{\mathcal{W}}\cos\beta \end{bmatrix}. \tag{3.65}$$

The Lagrangian can now be obtained using the total kinetic energy and potential energy of the system and the equations of motion can be solved using the Euler-Lagrangian equation. The resulting coupled differential equations can then be decoupled where desired. This process is automated by using the Symbolic Toolbox of MATLAB and the final equations are given by

$$\ddot{x}_q = -\frac{(c\dot{\beta} + k\beta + l f_{\mathcal{W}}\cos\beta)(m_q + m_p)\cos\beta + l m_q m_p \sin\beta(l\dot{\beta}^2 + g\cos\beta)}{l m_q (m_q + m_p)}, \tag{3.66}$$

$$\ddot{z}_q = -\frac{(c\dot{\beta} + k\beta + l f_{\mathcal{W}}\cos\beta)(m_q + m_p)\sin\beta - l m_q m_p \cos\beta(l\dot{\beta}^2 + g\cos\beta)}{l m_q (m_q + m_p)}, \tag{3.67}$$

$$\ddot{\beta} = \frac{l m_p\left(\ddot{x}_q\cos\beta + \ddot{z}_q\sin\beta - g\sin\beta\right) - c\dot{\beta} - k\beta - l f_{\mathcal{W}}\cos\beta}{l^2 m_p}, \text{ and} \tag{3.68}$$

$$f_{\mathcal{W}} = \frac{1}{2}\rho\left(-\dot{x}_p + V_{\mathcal{W}_X}\right)^2 C_{D_P}, \tag{3.69}$$

where C_{D_P} is the aerodynamic drag coefficient of the payload.

3D Quadrotor with Suspended Payload Forces and Moments

In this section, the equations of motion obtained in the previous section are used to augment the 2D payload model and the 3D quadrotor model. The equations of motion obtained in the previous section describe the horizontal and vertical accelerations of the quadrotor vehicle under the influence of the payload. Using Newton's second law of motion, the inertial force experienced by the quadrotor due to the payload is calculated as

$$\boldsymbol{F}_{\mathcal{I}}^{\mathrm{P}} = m_q \begin{bmatrix} \ddot{x}_q \\ 0 \\ \ddot{z}_q \end{bmatrix}. \tag{3.70}$$

The body force experienced by the quadrotor due to the payload is calculated by transforming the inertial force into the body frame.

$$\boldsymbol{F}_{\mathcal{B}}^{\mathrm{P}} = \boldsymbol{R}_{V}\boldsymbol{F}_{\mathcal{I}}^{\mathrm{P}}. \tag{3.71}$$

This concludes the derivation of the forces and moments that the suspended payload exerts on the quadrotor vehicle.

3.5 Summary

In this section the kinetics, kinematics and equations for the forces and moments are given for a quadrotor with a suspended payload. A block diagram showcasing the model described by the equations of motion is shown in Fig. 3.5.

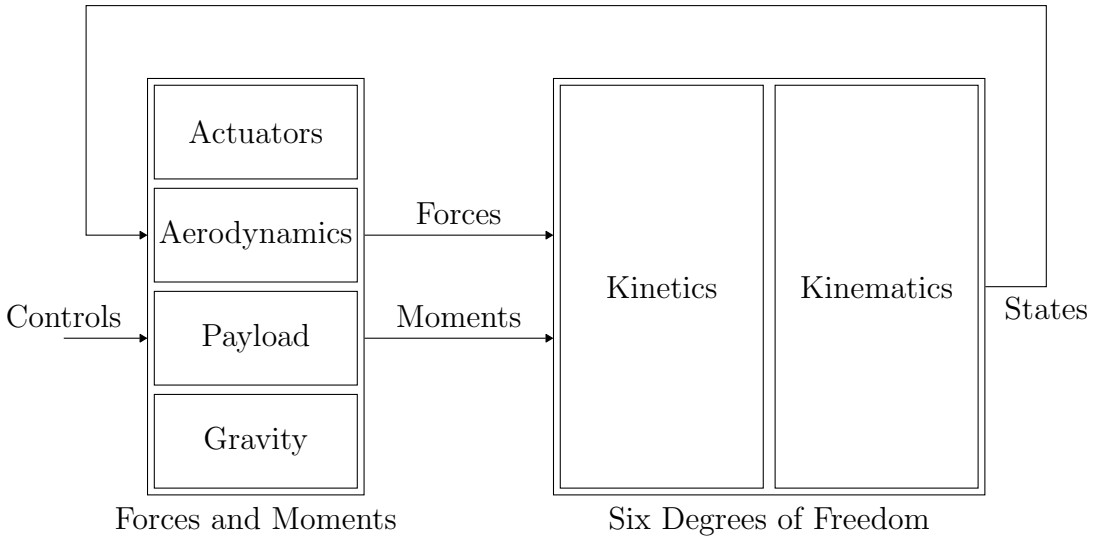


Figure 3.5: Quadrotor and payload model block diagram.

The model derived in this section can be used to simulate any quadrotor vehicle with a suspended payload, given its parameters. The mathematical model also provides insight of the system, aiding the design of the control system.

4. Hardware Design

This chapter describes the design procedure of the hardware used for this project. A detailed discussion of the chosen avionics and vehicle design is given with respect to the payload and flight time capabilities. This is followed by the procedure of obtaining all the physical properties of the vehicle to implement a non-linear model for simulation purposes.

4.1 Vehicle Hardware

A quadrotor is chosen as the vehicle for practical experiments. The quadrotor will carry an unknown suspended payload, which affects the flight dynamics of the vehicle. This requires the quadrotor to produce enough thrust to lift the payload and perform maneuvers for transportation purposes.

4.1.1 Quadrotor Components

A quadrotor has a lot of different components, which consist of the frame, propulsion system, power system and, avionics. The integration of these components are shown in Fig. 4.1 and the function of each is discussed in Table 4.1.

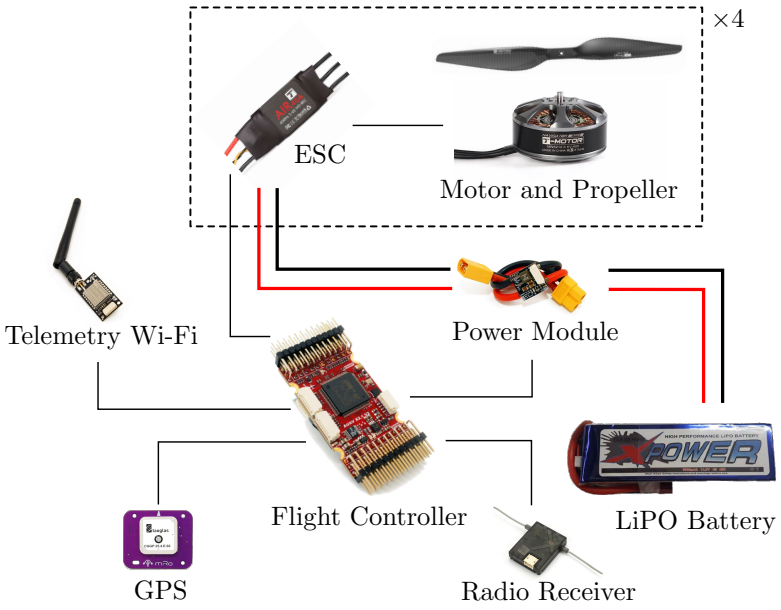


Figure 4.1: Diagram of the different components of a quadrotor.

Two custom components are included in the hardware of the vehicle, namely:

- A motor current measure module.
- A payload angle measure module.

Component	Function
Avionics	
Flight Controller	The flight controller receives measurements from sensors, estimates the states of the vehicle and commands an actuator signal accordingly. The flight controller has on-board sensors such as the Inertial Measurement Unit (IMU), barometer and magnetometer.
IMU	The IMU consists of a gyroscope and an accelorometer, measuring angular rates and acceleration. These measurements are used to estimate the attitude of the vehicle.
Barometer	The barometer measures pressure which can be used to estimate the vehicle's altitude.
Magnetometer	The magnetometer measures the magnetic field and is used to estimate the heading of the vehicle.
GPS	The Global Positioning System (GPS) provides measurements of the position and linear velocity of the vehicle.
Telemetry Wi-Fi	It provides a wireless connection to the groundstation computer to monitor the vehicle or send commands.
Radio Receiver	It provides a wireless connection to the transmitter for manual control of the vehicle.
Propulsion System	
ESC	An Electronic Speed Controller (ESC) receives a Pulse Width Modulation (PWM) signal and, depending on the duty cycle, controls the speed of the motor accordingly.
Motor and Propeller	It produces thrust to fly the vehicle.
Power System	
LiPo Battery	A Lithium Polymer (LiPo) battery is used for its high capacity and discharge rate.
Power Module	It passes the battery power to the ESCs, provides stable power for the avionics and provides voltage and current measurements of the battery.

Table 4.1: The function of each of the quadrotor's components.

The motor current measure module is a custom Printed Circuit Board (PCB), shown in Fig. 4.2, designed to measure the current draw of each individual motor. The purpose of this module is to obtain feedback on the performance of the motors. During hover, the current draw of each motor is expected to be similar and this module provides feedback to ensure that this is the case.

The payload angle measure module, shown in Fig. 4.3, consists of a potentiometer in a voltage divider circuit. The potentiometer rotates with the suspended payload and therefore, the voltage over the potentiometer provides a payload angle measurement. This module is used to log the payload angle for offline analysis.

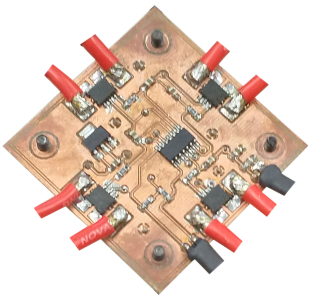


Figure 4.2: Motor current measure module.

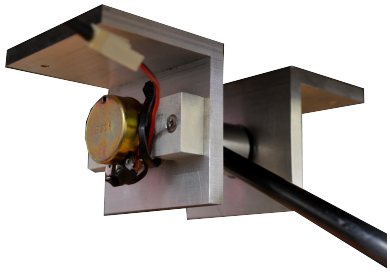


Figure 4.3: Payload angle measure module.

The design process of the hardware is an iterative one as the interactions of the different components are quite coupled, especially with the propulsion and power system. Hardware for the project was chosen using this iterative process. The avionics are discussed first, followed by the complete quadrotor design and capabilities.

Avionics

Different options for flight controllers exist. DJI is a popular company concerning multirotors and has a wide range of available vehicles and flight controllers. However, their products are closed systems and it is quite difficult to extend and modify. It was decided to rather work with open-source software.

The available open-source flight control software stacks are PX4, ArduPilot, Betaflight, and iNav. Each of these flight stacks are supported by specific hardware. Therefore, the flight stack is first chosen and then one of the supported hardware options are decided upon.

Figure 4.4: Open-source flight control software stacks [36, 37, 38, 39].

In short, PX4 and ArduPilot are both run by the same hardware and focus on autonomous flight of medium to large size vehicles. BetaFlight and iNav are generally both run by the same hardware. BetaFlight focusses on the hobbyist quadrotor racers and iNav focusses on autonomous flight of small to medium size vehicles. Therefore, the obvious choice of these flight control software stacks is either PX4 or ArduPilot.

Both PX4 and ArduPilot flight stacks were explored. PX4 makes use of an asynchronous structure to enable easy time scheduling between the different software components. ArduPilot requires the time scheduling to be done manually between most of the software components. Therefore, the structure of PX4 is easier to understand and the process of making modifications to the codebase is simpler for new developers. PX4 also supports both Software-in-the-Loop (SIL) and Hardware-in-the-Loop (HIL) simulations, whereas ArduPilot only supports SIL simulations. Therefore, it was decided to use PX4 as the firmware of the vehicle.

The most popular, reliable and supported flight controller for PX4 is the Pixhawk series. Pixhawk is an open-source hardware design for flight controllers containing a microcontroller, running the NuttX Real Time Operating System (RTOS), and on-board sensors such as an IMU, magnetometer, and barometer. The mRobotics Pixhawk X2.1, uBlox GPS, Wi-Fi telemetry module, and power module are the avionics of choice.

Final Quadrotor Design

The final quadrotor design with all of the chosen components are shown in Table 4.2. The final design is a large quadrotor with a motor-to-motor distance of 960 mm, a carbon fibre frame and a powerful propulsion system.

Component Type	Component Name	Weight (kg)	Quantity	Total Weight (kg)
Frame	Tarot X4 960mm	1.6	1	1.6
Autopilot	mRo X2.1 Rev 2	0.04	1	0.04
GPS	mRo GPS u-Blox Neo-M8N	0.01	1	0.01
Telemetry	mRo Wi-Fi Module ESP8266	0.01	1	0.01
Power Module	mRo Power Module ACSP7	0.02	1	0.02
Radio Receiver	Spektrum Satellite Receiver	0.01	1	0.01
Motor	T-Motor MN5212 KV340	0.249	4	0.996
ESC	T-Motor Air 40A ESC	0.026	4	0.104
Propeller	T-Motor P18x6.1	0.026	4	0.148
Battery	X-Power 6S1P 5000mAh 45C	0.7	2	1.4
		Total Weight		**4.338**
		Weight Safety Factor		5%
		Estimated Weight		**4.555**

Table 4.2: Designed quadrotor components.

The most important factors when designing a quadrotor are the total mass of the vehicle, the current draw of the propulsion system and the capacity of the power system. These factors determine the total flight time and payload capabilities. A weight safety factor of 5% is included in the design to account for the weight of the wires, screws, bolts, etc.

The payload capabilities of the designed quadrotor is shown in Table 4.3. The calculations of these values are given in Appendix A.2.

	Payload Mass	Flight Time
Maximum Payload Capacity	4.155 kg	12 min 40 s
No Payload	0 kg	25 min 13 s

Table 4.3: Quadrotor payload capabilities.

This design is capable of carrying a large payload and provides ample flight time for practical flight tests where the controllers can be evaluated. The vehicle was built and the complete quadrotor is shown in Fig. 4.5. The unknown parameters within Chapter 3 can now be measured to increase the accuracy of the non-linear model for use in simulations.

Figure 4.5: Custom built quadrotor with a suspended payload.

4.2 Quadrotor Physical Parameters

The easiest parameters to measure are the mass of the vehicle and the distance from the motors to the center of the quadrotor. The measured mass of the vehicle is 4.5 kg, which is close to the calculated mass of the design process. The distance from the center of the vehicle to a motor is measured as 0.49 m.

The other physical parameters are obtained through experiments discussed in the following sections.

4.2.1 Mass Moment of Inertia

The mass moment of inertia is not easy to calculate for arbitrary objects. Two methods are used to determine the mass moment of inertia of the quadrotor. The first method is to calculate an estimate of the values mathematically. The second method is to perform an experiment to determine the values.

Mathematical Calculation

The mass moment of inertia for several common shapes are easy to determine. Therefore, the quadrotor physical model is simplified, as seen in Fig. 4.6.

Only the main components that make a significant contribution to the mass moment of inertia are considered. These components are simplified and the masses are lumped together. After calculating the mass moment of inertia of each component, the parallel axis theorom is used to calculate the inertia about the axes of the vehicle.

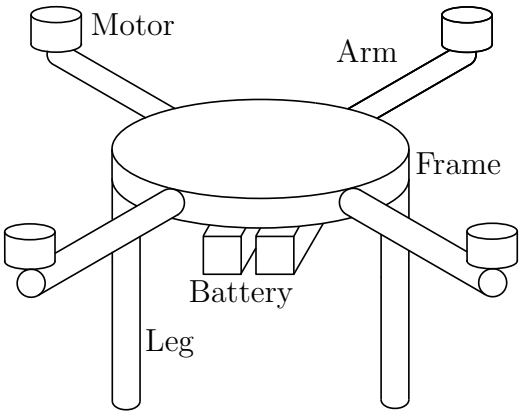


Figure 4.6: Simplified quadrotor model

The estimated mass moment of inertia about each axis is shown in Table 4.4.

Inertia	$\boldsymbol{I}$ **(kgm^2)**
I_{xx}	0.2231
I_{yy}	0.2212
I_{zz}	0.3022

Table 4.4: Mass moment of inertia mathematical results

Experimental Analysis

An experiment was also conducted to determine the mass moment of inertia, based on previous work done by [40]. The experiment is known as the two rope inertia experiment. The experiment is explained in detail in Appendix A.3 and the final results are shown in Table 4.5.

Inertia	I (kgm^2)
I_{xx}	0.230
I_{yy}	0.235
I_{zz}	0.328

Table 4.5: Mass moment of inertia experimental results

Discussion

It is clear that both the mathematical and experimental generated results are relatively close. The mathematical method makes use of a lot of simplifications and assumptions of the structure of the vehicle that are not necessarily true, therefore it is expected that it will not yield an accurate result. The mass moment of inertia results obtained from the experiment are considered as the true values and is used in the non-linear model and during the controller design process.

4.2.2 Thrust vs PWM Mapping

To setup an accurate motor model, the mapping from the input PWM signal to the output thrust of each motor is required. A thrust test jig was used to accomplish this, which consists of a load cell to measure the output thrust of the motor and logs the data. The Pixhawk was used to send step inputs to the ESC. This was done for every motor and the results are shown in Fig. 4.7.

The measured motor thrust is close to the manufacturer supplied motor performance data, given in Table A.1. However, one of the motors is underperforming in relation to the rest. It is desirable to have all of the motors' performance equal at hover thrust, for smooth takeoff and minimal correction from the controllers during hover. Therefore, the motor commands are normalized at hover, without any payload. It was found that the input PWM pulse width for hover is 1350μs. Third-order polynomials are fitted to the raw data to obtain functions for each motor from the input PWM signal to the output thrust. The normalized fitted functions are shown in Fig. 4.8.

The mathematical normalized functions are described in Appendix A.4.

4.2.3 Motor Time Constant

When a motor is given an input to run at a specific speed, it takes some time to spin up and achieve the desired speed. This time can be described by a time constant, which is the time it takes for the motor to achieve 67% of the desired speed. This parameter is needed as it defines the maximum speed of the physical system.

The time constant depends on a number of factors such as the input voltage, current draw, input step size, etc. Therefore, the time constant changes depending on these factors. A lot

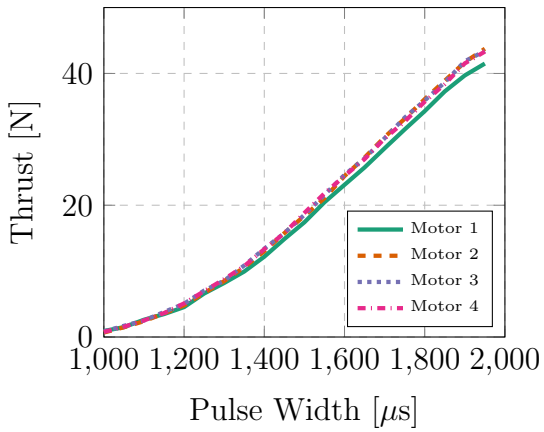


Figure 4.7: The raw PWM and thrust relation of each motor.

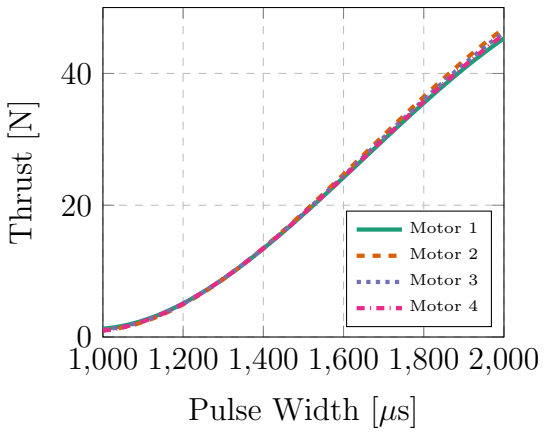


Figure 4.8: The normalized and fitted PWM and thrust relation of each motor.

of research has been done on electric brushless motors and, as mentioned in Chapter 3, a first-order model is generally chosen to model such a motor without depending on these factors. The appropriate time constant should be chosen with care. If the time constant is chosen too fast, there is a risk of designing controllers that respond faster than the physical system. Because of this, a conservative time constant was chosen. Larger step inputs were given as this results in slower time constants. One of the step responses of the aforementioned thrust experiment is shown in Fig. 4.9. It is difficult to analyze the data and obtain a time constant as the signals are noisy and the time constant varies on each step. It was found that the time constant, on average, for the steps is $\tau = 0.07$ s.

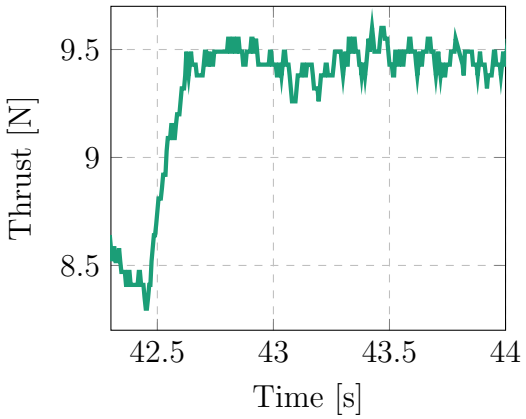


Figure 4.9: A Motor Step Response

4.2.4 Virtual Yaw Moment Arm

The virtual yaw moment arm is a coefficient that relates thrust to torque, as shown in the equation

$$\tau_i = R_N T_i, \tag{4.1}$$

where τ_i is the torque of the i_{th} motor, T_i is the thrust of the i_{th} motor and R_N is the virtual yaw moment arm coefficient. This coefficient is used to calculate the moment about the $\boldsymbol{z}_{\mathcal{B}}$ axis of the vehicle, as seen in Eq. (3.33).

Three different methods are used to calculate an estimated value of the virtual yaw moment arm coefficient. The first method referred to as the rotor drag point force method, assumes that the drag force is applied at one single point on the rotor. The second method makes use of blade element theory and the last method is an experimental method where measurements of the thrust and torque are taken. These methods are discussed in detail in Appendix A.5. The estimated virtual yaw moment arm coefficient calculated by each method is summarized in Table 4.6.

Method	**Coefficient** R_N
Rotor Drag Point Force	0.0180 m
Blade Element Theory	0.0237 m
Experiment	0.0218 m

Table 4.6: Calculated values of the virtual yaw moment arm coefficient.

The different values obtained from the methods are all in the same order of magnitude and the result from the experimental method is chosen as the final virtual yaw moment arm.

4.2.5 Aerodynamic Coefficients

The aerodynamic coefficients of the vehicle are determined by performing a practical flight experiment in no-wind conditions.

The experiment involves flying the vehicle in either the $\bar{\boldsymbol{x}}_{\mathcal{I}}$, $\bar{\boldsymbol{y}}_{\mathcal{I}}$ or $\bar{\boldsymbol{z}}_{\mathcal{I}}$ direction at a constant velocity, as shown in Fig. 4.10. During this maneuver, the attitude of the vehicle, θ, and the forces experienced by the vehicle due to the actuators, f_T, and gravity, f_g, are known. The only unknown force is that caused by the aerodynamic drag, $f_{\mathcal{W}_X}$. This force can be calculated by using Newton's second law of motion, and because the vehicle is flying at a constant velocity, the acceleration is zero.

After obtaining the aerodynamic drag force in a specific direction, the Eq. (3.45) can be used to calculate the aerodynamic coefficient of the specified axis. The aerodynamic drag coefficient in each of the axes was found to be 0.2 m^2.

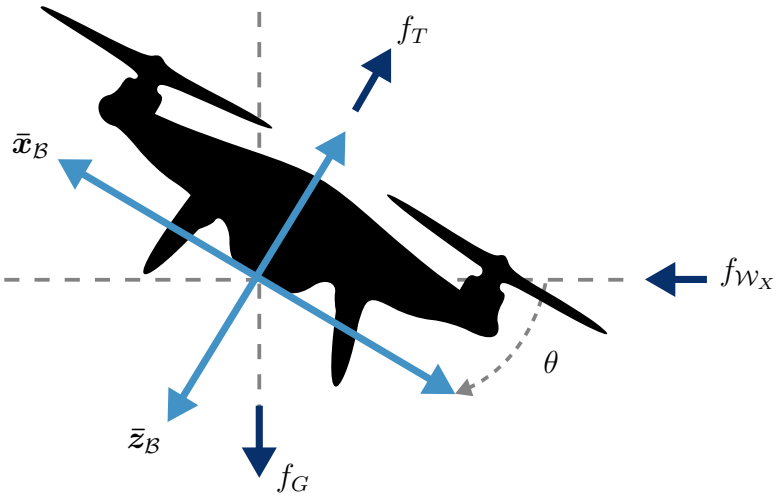


Figure 4.10: Qaudrotor flying at a constant velocity.

4.3 Summary

A summary of all the physical parameters of the vehicle are given in Table 4.7.

Parameter	**Value**
Mass (m)	4.5 kg
Mass Moment of Inertia about $\bar{\boldsymbol{x}}_{\mathcal{B}}$ (I_{xx})	0.23 kgm^2
Mass Moment of Inertia about $\bar{\boldsymbol{y}}_{\mathcal{B}}$ (I_{yy})	0.235 kgm^2
Mass Moment of Inertia about $\bar{\boldsymbol{z}}_{\mathcal{B}}$ (I_{zz})	0.328 kgm^2
Motor Distance (d)	0.49 m
Virtual Yaw Moment Arm (R_N)	0.0218 m
Motor Time Constant (τ)	0.07 s
Body Aerodynamic Coefficient about $\bar{\boldsymbol{x}}_{\mathcal{B}}$ (C_{D_X})	0.2 m^2
Body Aerodynamic Coefficient about $\bar{\boldsymbol{y}}_{\mathcal{B}}$ (C_{D_Y})	0.2 m^2
Body Aerodynamic Coefficient about $\bar{\boldsymbol{z}}_{\mathcal{B}}$ (C_{D_Z})	0.2 m^2

Table 4.7: Physical parameter values of the quadrotor

This chapter discussed the design process of the physical vehicle and the methods of obtaining the physical parameters. All the parameters needed to simulate the physical vehicle with the differential equations, derived in Chapter 3, were obtained.

5. Flight Control Toolchain Overview

In this section, the flight control toolchain is discussed. The toolchain mainly consists of PX4, Gazebo, and the Robot Operating System (ROS). PX4 is the flight control firmware of the vehicle, Gazebo is a simulation environment and ROS is used to command the vehicle.

5.1 PX4 Overview

PX4 was created at the same time as the Pixhawk. It was the intended firmware for the Pixhawk boards. The flight control software development began with a group of students from ETH Zürich. The open-source project has grown tremendously and is being used by various companies in the industry, such as Auterion. The latest stable release of PX4 at the time of this project is 1.9.2, which is the version used for this project.

The PX4 codebase consists of various components, called modules. Each module is dedicated to a specific task. A block diagram of the structure of PX4 is shown in Fig. 5.1 [41]. Each block consists of various modules.

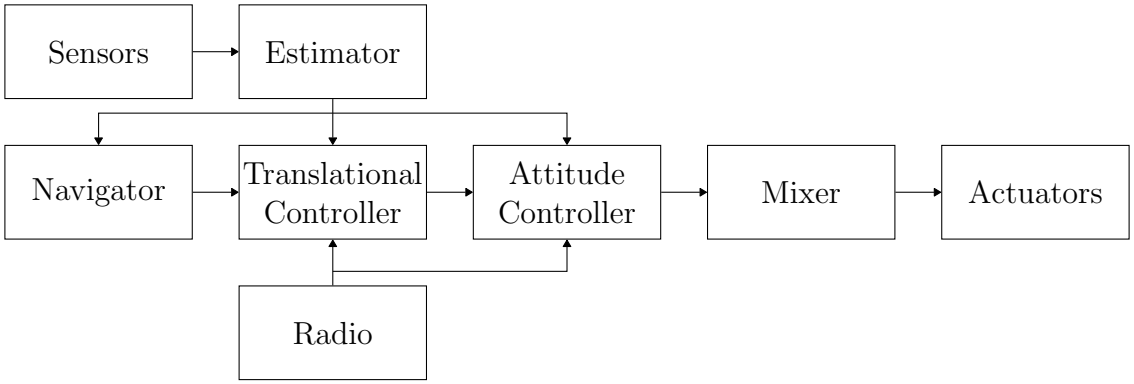


Figure 5.1: PX4 building blocks.

5.1.1 Architecture

The different modules communicate with each other by means of a publish-subscribe architecture, which is an asynchronous message passing system. This allows the modules to publish messages to a specific topic that other modules can subscribe to, to receive the messages.

The asynchronous architecture makes the design quite modular. Any module can be exchanged with a different one and as long as it adheres to the specific topic subscriptions and publications, it will run as intended. This allows developers to easily make modifications to the codebase.

PX4 supports various hardware options, such as the NuttX RTOS, running on the Pixhawk, and Linux. The ability to build PX4 for Linux is beneficial as it allows developers to run simulations on a Linux desktop with the exact code that will run on the target hardware of the vehicle.

5.1.2 Estimator

The estimator used by PX4 is an Extended Kalman Filter (EKF). The EKF receives sensor measurements and combines them to compute the estimated state of the vehicle. The estimator makes use of a delayed time horizon architecture to account for the different sample times of the sensors. Typically, the estimator runs at 1 kHz and publishes the estimated states at 250 Hz.

5.1.3 Controllers

PX4 uses classical linear PID controllers for the control of the vehicle. The control architecture is based on a cascaded loop structure. The controllers consist of the inner attitude and outer translational controllers. The inner attitude controllers run at 250 Hz, which is the rate at which the estimated states are received from the EKF. The outer translational controllers run at 50 Hz. The controllers are discussed in depth in Chapter 6.

5.1.4 Simulation

PX4 makes use of the Gazebo simulator and can perform both SIL and HIL simulations. SIL is a simulation environment running both the non-linear model and the flight controllers on the development machine. HIL is a simulation environment running the non-linear model on the development machine and the flight controllers on the target hardware with a communication link between them. SIL is used to test algorithms in the development stage and HIL is used to test the algorithms on the target hardware, making sure it can run with constrained resources.

The HIL simulation environment is necessary for this project as the flight controllers will be modified and need to be tested on the Pixhawk hardware before a flight.

5.2 PX4 Modifications

The following modifications were made to PX4 for this project:

- A custom module was implemented to log the individual currents drawn from each motor, as discussed in Section 4.1.1.
- A custom module was implemented to log the swing angle of the suspended payload, as discussed in Section 4.1.1.
- The PWM scaling factors used to normalize the thrust at hover, as described in Section 4.2.2, was implemented.
- A custom module to provide step inputs to the motors for identification purposes, which was used in Section 4.2.2, was implemented.

These modifications were implemented on the PX4 firmware version 1.9.2 to create a custom firmware version. The custom version is flashed to the Pixhawk for flight.

In the subsequent chapters, the controllers of PX4 are modified to accommodate the unknown suspended payload.

5.3 Simulation Environment

PX4 is developed in the C++ programming language. To design and test algorithms in this environment can produce slow development cycles. Therefore, it was decided to make use of MATLAB/Simulink, to design and test algorithms. When the design process is finished in MATLAB/Simulink, the algorithm will be implemented in C++ in the PX4 environment and tested in Gazebo via a SIL simulation. Once this is successful, the modified PX4 firmware is flashed to the Pixhawk on the quadrotor and verified with a HIL simulation. After this process, the quadrotor can be used during a practical flight test to verify the algorithm.

This is the development cycle used during this project. There are two simulation environments, namely MATLAB/Simulink and the PX4-Gazebo environment. These simulation environments are illustrated in Fig. 5.2 and the practical setup is illustrated in Fig. 5.3.

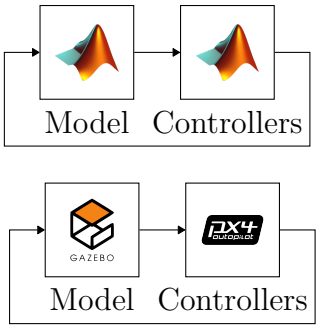


Figure 5.2: Block diagrams of the simulation environments.

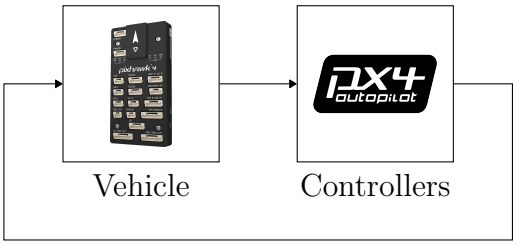


Figure 5.3: Block diagram of the practical setup.

5.3.1 MATLAB/Simulink Simulation

The MATLAB/Simulink environment includes both the non-linear model and flight controllers.

The non-linear model is implemented by the differential equations derived in Chapter 3. It includes simple noise models for the states, which includes high-frequency noise and a sensor bias.

A detailed analysis of the PX4 flight control system was conducted and the flight controllers are presented in Chapter 6. A simplified version of these controllers is implemented in MATLAB/Simulink for the simulation, which makes for faster development.

5.3.2 Gazebo Simulator

Gazebo is a robotics simulation environment. It consists of two main programs, namely gzserver and gzclient. The gzserver program is a physics engine and is responsible for all the physics calculations. The gzclient program is a graphical renderer responsible for the visualization of the simulation.

Gazebo is a powerful simulation environment capable of modeling objects, modeling the environment, adding sensors and applying forces and moments.

Gazebo makes use of graphical modeling. Graphical modeling describes an object using a variety of links and joints. Therefore, the differential equations describing a model, such as

those in Chapter 3, are not implemented directly. Rather, the object is described by links and joints and the physics engine solves the differential equations describing the interaction of these links and joints. The model is described by the Simulation Description Format (SDF), which is an Extensible Markup Language (XML) format describing robot environments and objects.

Sensors such as an IMU, GPS, camera or LiDAR, to name but a few, can be added to the model. These sensors are used to simulate the system with data that is more realistic. The noise models implemented by PX4 for the Gazebo simulation environment includes high-frequency noise, sensor bias, and low-frequency random walks. These noise models can be adjusted to act like a specific real-world sensor.

Gazebo provides a mechanism to interact with the objects and environment, called plugins. A plugin is a custom piece of code, capable of extracting information from the simulation or interacting with the objects therein.

All of these features make Gazebo an ideal candidate for a simulation environment. Given that a communication protocol between PX4 and Gazebo exists, it makes sense to use Gazebo as the simulation environment for SIL and HIL simulations. The SIL architecture is shown in Fig. 5.4 and the HIL architecture is shown in Fig. 5.5.

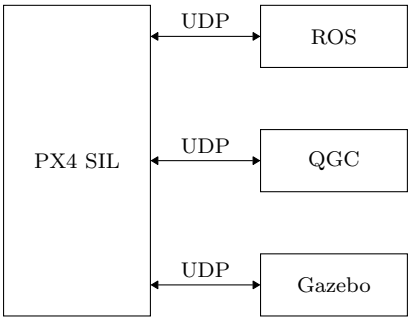


Figure 5.4: PX4 SIL Gazebo communication

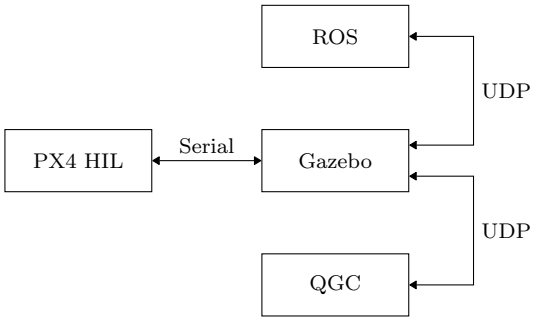


Figure 5.5: PX4 HIL Gazebo communication

The physical quadrotor and payload system, introduced in Chapter 4, is modelled in Gazebo, as shown in Fig. 5.6. The Gazebo physics engine is capable of modeling the six degrees of freedom quadrotor model, the effect of gravity and the coupling between the quadrotor and the payload. Custom plugins were written to apply the forces and moments of the actuators and aerodynamic drag, derived in Chapter 3.

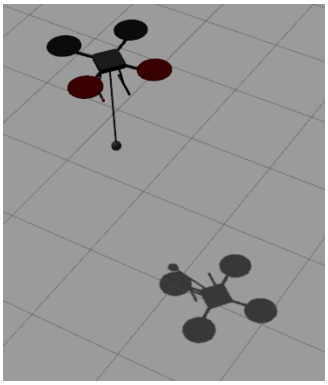


Figure 5.6: Custom built quadrotor Gazebo model.

5.3.3 Discussion

The implementation of these two simulation environments are vastly different. The MATLAB/Simulink environment implements the differential equations of the model, described in Chapter 3, with all the states available for measurement and makes use of a simplified version of the PX4 controllers. The Gazebo environment makes use of graphical modeling and a physics engine for the model and the full PX4 flight control stack with estimation and control.

Considering all of these differences, it is important to be vigilant when migrating from one environment to the other. Therefore, the two environments are compared to ensure that, for a given scenario, the response is nearly identical. The comparison is described and results are shown in Chapter 6.

5.4 Practical Flight Setup

An illustration of the practical flight setup is shown in Fig. 5.7. The setup consists of the quadrotor vehicle, the radio transmitter for manual control and a ground station computer running both QGroundControl (QGC) and ROS.

5.4.1 QGroundControl

The ground control software used by the PX4 stack is QGC. A wireless link is established between the vehicle and the ground station computer running QGC, in this case via Wi-Fi.

QGC is used during the setup of the vehicle to guide the user through the sensor calibration progress and setting parameters for the specific vehicle. During flight, QGC is used to command the vehicle during autonomous flight. The vehicle can be commanded to autonomously take off, fly through a set of waypoints, return to the home position and autonomously land. QGC is an integral part of the PX4 flight stack regarding autonomous flight.

QGC also provides feedback of the state of the vehicle. It allows the pilot to monitor the vehicle, making sure the flight is going according to plan.

In this project, QGC is mainly used for the setup of the vehicle and to monitor the vehicle during flight.

Figure 5.7: Practical flight setup illustration.

5.4.2 ROS

ROS is used for external control of the vehicle. In this case, it is used where the functionality of QGC lacks the desired control. A brief overview of ROS is given below, followed by how it is used in this project.

ROS is the Robot Operating System. However, unlike the name suggests, it is not an operating system but rather runs on top of Linux. ROS is a collection of tools and libraries commonly used for robotics applications. At its core, it is basically just a communication infrastructure.

An individual ROS component is called a ROS node and is generally given a single task. Therefore, various ROS nodes are used collectively in a system to achieve the desired outcome. ROS primarily makes use of a publish-subscribe architecture, as shown in Fig. 5.8, for the communication between nodes. A ROS master node handles the communication and every other ROS node registers with the master before it can communicate with other nodes.

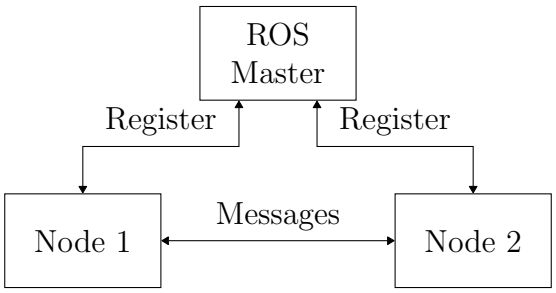


Figure 5.8: ROS communication setup

ROS has grown to become a standard tool in robotics applications. It has an active community that provides standard ROS nodes for applications such as path planning, computer vision, localization, mapping, etc. This wide collection makes ROS a popular solution in the robotics field.

In this project, ROS is used to set up repeatable flight tests. QGC can be used to command waypoints using GPS coordinates, but this requires the same location and takeoff position for every flight, which renders this approach non-repeatable. On the other hand, waypoints can be sent with ROS in the local NED coordinate frame, enabling repeatable flight tests no matter the location. Two custom ROS nodes are used in this project, namely a step input node and a waypoint scheduler node.

Step Input ROS Node

The step input ROS node provides a way to generate angular rate, angle, linear velocity, and position step inputs. It is used in simulation to verify whether the controllers meet the design requirements, during the design of the controllers in Chapter 6. The architecture of this setup is shown in Fig. 5.9.

Waypoint Scheduler ROS Node

The waypoint scheduler ROS node is used to generate NED waypoints for the vehicle to follow in a practical flight. The architecture of this setup is shown in Fig. 5.10.

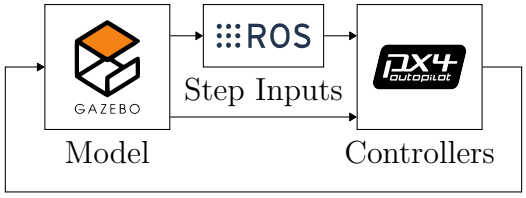


Figure 5.9: ROS simulation architecture diagram

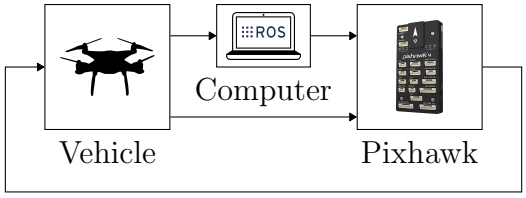


Figure 5.10: ROS practical flight architecture diagram

5.5 Summary

This chapter discussed the software tools needed for simulation and practical flights. The different components of this toolchain are summarized in Table 5.1.

Toolchain Component	**Description**
PX4	The flight control software stack including the estimator and controllers.
Gazebo	The simulation environment capable of running SIL and HIL simulations with the PX4 stack.
QGroundControl	The ground control software to setup, monitor and command the vehicle.
ROS	ROS is used for high-level control of the system.

Table 5.1: Summary of the toolchain components.

Modifications were made to the PX4 stack for the purpose of this project and a model of the custom built quadrotor and suspended payload was implemented in the Gazebo environment. QGroundControl is used during the setup of the vehicle and to monitor the state of the vehicle during flight. ROS is used to send external reference commands to the quadrotor during simulation or flight.

After the identification and investigation of these tools, they can now be used and modified for the purposes of this project.

6. Vehicle Control System

The focus of this chapter is the design of the control system for the quadrotor vehicle, without a payload. The control system is tested within simulation and demonstrated in a practical flight test. Only after the controllers have proved to maintain stable flight, will the suspended payload be considered. The control system design of the quadrotor with the suspended payload is conducted in Chapter 7.

This chapter investigates the control system architecture of PX4. The control system is designed according to this architecture. Thereafter, the designed control system is tested in simulation. The simulation results of the MATLAB/Simulink and PX4-Gazebo environments are compared. Finally, the control system is demonstrated in a practical flight test.

6.1 Control System Design

The control system architecture used in the PX4 stack is identified. The control gains for the quadrotor vehicle are designed according to this architecture. Only the longitudinal controllers are presented in this chapter to avoid unnecessary duplication. The controller gains and the design of the lateral, heave and directional controllers are described in Appendix B.

6.1.1 Control System Design Strategy

Classical control theory is used to design the control gains for the practical quadrotor. The control system design is based on a linear plant. The model derived in Chapter 3 is linearised around hover and the small-angle approximation is applied. The linearization process is fully described in [40].

The controllers are implemented on a microprocessor and, therefore, by definition they are discrete controllers. The emulation discretization method is used during the control design. Therefore, continuous controllers are designed and discretized to be implemented. However, the sampling rate of the controllers is much higher than the bandwidth of the vehicle dynamics. It was decided not to discretize the controllers and implement the continuous-time control gains directly as this separation is large enough.

The continuous-time controllers are designed with the root locus method. The inner-most controller is designed first, followed by the design of the outer controllers. To ensure a good time-scale separation between the controllers, the bandwidth of a controller should be at least 2 times slower than that of its inner controller. The control system architecture of PX4 is used, for which the control gains of the quadrotor are determined.

6.1.2 PX4 Control System Architecture

PX4 makes use of a cascaded loop architecture, consisting of linear PID controllers. The controllers consist of two categories, namely the inner attitude controllers and the outer trans-

lational controllers. The attitude controllers operate in the body frame and consist of the angular rate and angle controllers. The translational controllers operate in the inertial frame and consist of the linear velocity and position controllers. The control system architecture is shown in Fig. 6.1.

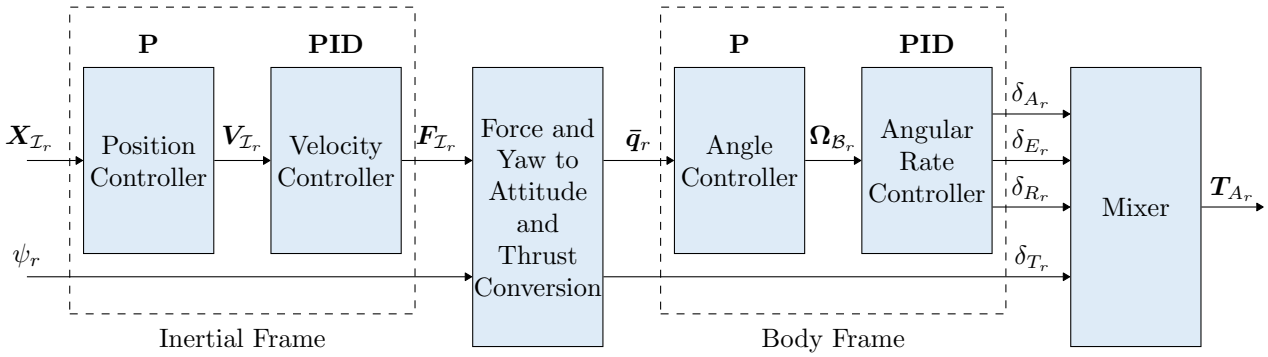


Figure 6.1: PX4 control system architecture.

The description of the symbols depicted in Fig. 6.1 are given as:

$\boldsymbol{X}_{\mathcal{I}_r}$	Inertial Position Reference
ψ_r	Yaw Reference
$\boldsymbol{V}_{\mathcal{I}_r}$	Inertial Linear Velocity Reference
$\boldsymbol{F}_{\mathcal{I}_r}$	Inertial Force Reference
$\bar{\boldsymbol{q}}_r$	Unit Quaternion Reference
$\boldsymbol{\Omega}_{\mathcal{B}_r}$	Body Angular Rate Reference
δ_{A_r}	Aileron Reference
δ_{E_r}	Elevator Reference
δ_{R_r}	Rudder Reference
δ_{T_r}	Total Thrust Reference
$\boldsymbol{T}_{A_r}$	Actuator Input Reference

In the following sections, each of these components are discussed and the design of the corresponding longitudinal controller is described.

6.1.3 Mixer

The mixer implements a mixing matrix for a specific vehicle. It converts the virtual actuator commands to actual actuator commands. After analyzing the codebase of PX4, the implemented mixing matrix for a quadrotor in the X-configuration, is found to be

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}_{\text{PX4}} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 1 \\ 1 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -1 \\ 1 & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -1 \end{bmatrix} \begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix}_{\text{PX4}}, \tag{6.1}$$

which is written in the compact form

$$\boldsymbol{T}_{A_{\text{PX4}}} = \boldsymbol{K}_{M_{\text{PX4}}}^{-1} \boldsymbol{\delta}_{V_{\text{PX4}}}. \tag{6.2}$$

This is a simplified representation of the PX4 mixing matrix implementation, as it also contains non-linearities, such as checking for actuator saturation and prioritising roll and pitch maneuvers over yaw maneuvers in the case of large actuator commands.

PX4 makes use of normalized actuator thrusts. The virtual actuators $\delta_{A_{\text{PX4}}}$, $\delta_{E_{\text{PX4}}}$ and $\delta_{R_{\text{PX4}}}$ are normalized to $[-1,\ 1]$, and the virtual actuator $\delta_{T_{\text{PX4}}}$ and the motor actuators $\boldsymbol{T}_{A_{\text{PX4}}}$ are normalized to $[0,\ 1]$. The derived equations in Section 3.4.1 defines the actuator thrust in Newtons. The maximum thrust of a motor, in Newtons, is defined as T_{MAX} and therefore, the thrust produced by a motor is limited to

$$0 \leq T_i \leq T_{\text{MAX}}, \tag{6.3}$$

where $i = \{1, 2, 3, 4\}$. The actuators are related to the normalized actuators by

$$\boldsymbol{T}_A = T_{\text{MAX}} \boldsymbol{T}_{A_{\text{PX4}}}. \tag{6.4}$$

Notice that the implemented mixing matrix differ from the one derived in Section 3.4.1. The derived mixing matrix is restated for convience as

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = 4 \begin{bmatrix} 1 & -\sqrt{2} & \sqrt{2} & 1 \\ 1 & \sqrt{2} & -\sqrt{2} & 1 \\ 1 & \sqrt{2} & \sqrt{2} & -1 \\ 1 & -\sqrt{2} & -\sqrt{2} & -1 \end{bmatrix} \begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix}, \tag{6.5}$$

which is written in the compact form

$$\boldsymbol{T}_A = \boldsymbol{K}_M^{-1} \boldsymbol{\delta}_V. \tag{6.6}$$

From Eq. (6.4), the relation between the virtual actuators and normalized virtual actuators is derived as

$$\begin{aligned} \boldsymbol{K}_M^{-1} \boldsymbol{\delta}_V &= T_{\text{MAX}} \boldsymbol{K}_{M_{\text{PX4}}}^{-1} \boldsymbol{\delta}_{V_{\text{PX4}}}, \\ \boldsymbol{\delta}_V &= T_{\text{MAX}} \boldsymbol{K}_M \boldsymbol{K}_{M_{\text{PX4}}}^{-1} \boldsymbol{\delta}_{V_{\text{PX4}}}, \\ \begin{bmatrix} \delta_T \\ \delta_A \\ \delta_E \\ \delta_R \end{bmatrix} &= T_{\text{MAX}} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \begin{bmatrix} \delta_{T_{\text{PX4}}} \\ \delta_{A_{\text{PX4}}} \\ \delta_{E_{\text{PX4}}} \\ \delta_{R_{\text{PX4}}} \end{bmatrix}. \end{aligned} \tag{6.7}$$

This relation needs to be taken into account when designing the controller gains for the derived model.

6.1.4 Angular Rate Controllers

Linear PID control is used to allow the vehicle to follow a given angular rate input $\Omega_{\mathcal{B}_{i,r}}$, where $i = \{X, Y, Z\}$. The controllers command the virtual actuators, δ_{k_r} where $k = \{A, E, R\}$, of the vehicle. A simplified block diagram of the implemented controller $D_{\Omega_i}(s)$ is shown in Fig. 6.2.

Standard PID control is implemented with the improvements listed in Table 6.1. This improves the robustness of the implemented control law and will not be included during the design phase.

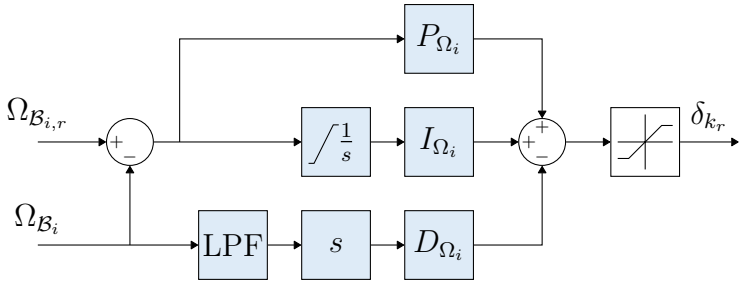


Figure 6.2: The implemented PX4 angular rate controller.

Improvement	**Motivation**
A Low Pass Filter (LPF) is added to the derivative path.	This reduces the effect of noise. Taking the derivative of a noisy signal will result in the amplification of the noise, which is an unwanted effect for control purposes.
The derivative path acts on the output of the plant.	This eliminates a phenomenon called derivative kick, described in [42].
The integral term is limited by an upper and lower bound.	This eliminates an effect known as integrator wind-up.
The control signal is limited by an upper and lower bound.	This eliminates actuator saturation in the case of large reference commands.

Table 6.1: Standard PID control improvements.

The block diagram shown in Fig. 6.2 is simplified and the implemented angular rate controller also contains

- a feedforward term used during trajectory tracking,
- a technique called Throttle PID Attenuation (TPA), which reduces the effect of the Proportional gain at high throttle values, but this is usually only used for racing drones, and
- a reduction of the Integral gain for fast angular rates, usually from doing flips to eliminate bounce-back.

These effects are not included as they are not relevant to this project.

The linear plant for the pitch rate of the derived model in Chapter 3, is given as

$$G_{\Omega_Y}(s) = \frac{\Omega_Y(s)}{\delta_{E_r}(s)} = \frac{\frac{d}{\tau I_{yy}}}{s\left(s + \frac{1}{\tau}\right)}, \tag{6.8}$$

where d is the distance from a motor to the CoM of the vehicle, I_{yy} is the mass moment of inertia about the y-axis, and τ is the time constant of the motors. The controller gains need to be designed for the normalized actuators. Therefore, from Eq. (6.7), the plant becomes

$$G^{\text{PX4}}_{\Omega_Y}(s) = \frac{\Omega_Y(s)}{\delta^{\text{PX4}}_{E_r}(s)} = \frac{2T_{\text{MAX}}\frac{d}{\tau I_{yy}}}{s\left(s + \frac{1}{\tau}\right)}. \tag{6.9}$$

A block diagram of the linear system used for the controller design is shown in Fig. 6.3. The root locus plot of the plant is shown in Fig. 6.4 and the root locus plot of the combined plant and proposed PID controller is shown in Fig. 6.5.

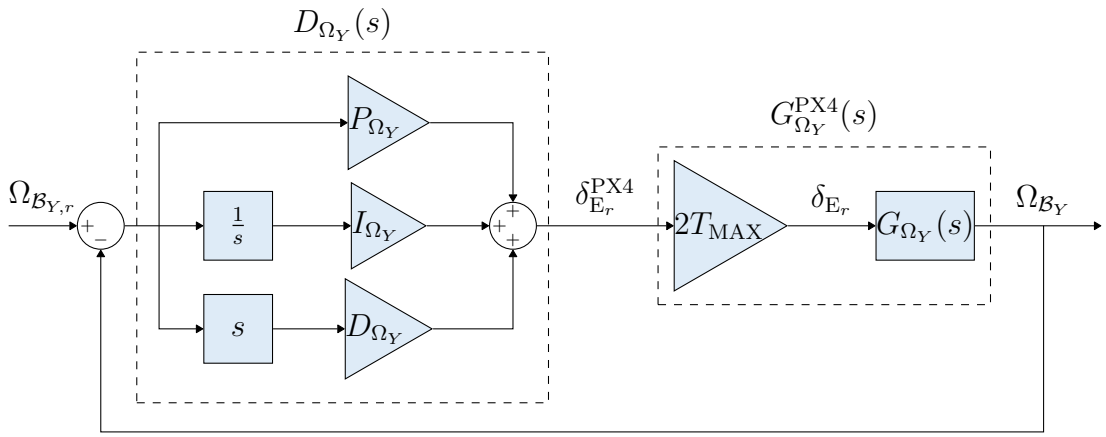


Figure 6.3: Pitch rate controller design block diagram.

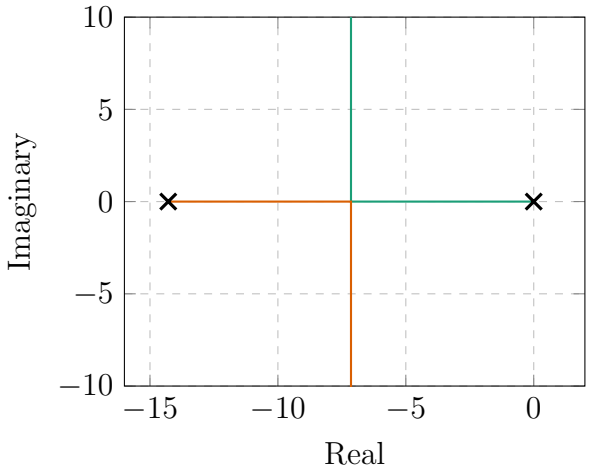


Figure 6.4: Root locus of the pitch rate dynamics.

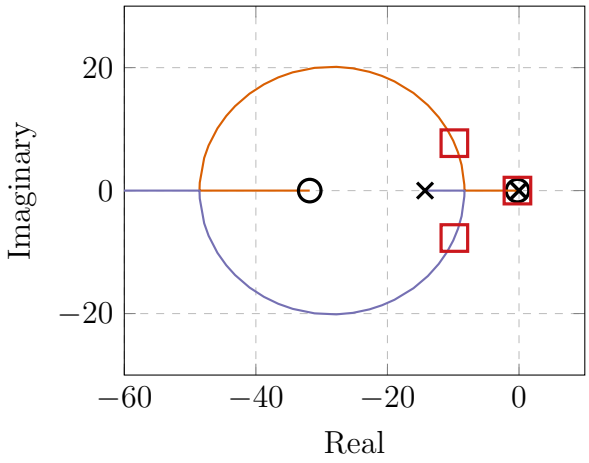


Figure 6.5: Root locus of the pitch rate controller.

The plant contains a pole at the origin and a pole corresponding to the actuator time constant. The pitch rate controller should

- have a fast response as it is a critical component regarding the stability of the system
- have a well-damped response
- reject steady-state errors due to effects such as mass inbalance and wind

The controller design starts with the proportional term to obtain the desired bandwidth. The integral term is added to reject disturbance torques. The system now has two free integrators, one from the plant dynamics and one from the controller, making it a type 2 system capable of tracking ramp inputs without any steady-state error. The integral term features a pole at the origin and a zero close to the origin at $s = -0.234$. Three closed-loop poles are obtained consisting of two underdamped poles and one pole close to the origin, which effect is minimized due to pole-zero cancellation. The two underdamped poles are, therefore, considered to be the dominant poles. Lastly, the derivative term is added to introduce more damping into the system. The result is a response with an overshoot of 4.2%, a bandwidth of 12.52 rad/s and a 2% settling time of 0.6 s.

The step response of the effect of the P, PI and PID terms of the pitch rate controller is shown in Fig. 6.6. The disturbance rejection that the integrator term offers is show in Fig. 6.7, where a constant disturbance torque is introduced at time $t = 2$ s.

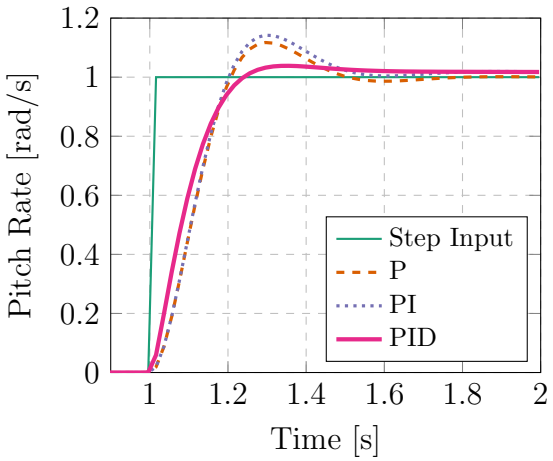


Figure 6.6: Step response of the pitch rate controller.

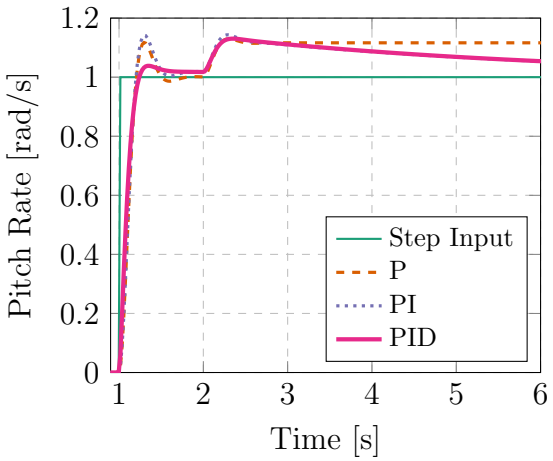


Figure 6.7: Disturbance rejection of the pitch rate controller.

The slow decay to a zero steady-state error, is a side-effect of the integrator term. The integrator gain can be reduced to minimize this long tail, but at the cost of disturbance rejection performance. Likewise, increasing the integrator gain results in better disturbance rejection, but results in a larger overshoot and longer tail. A trade-off was made in favor of a more damped response, but at the cost of slower disturbance rejection. The effect of the long tail can be minimized with pre-filtering [43], but it was decided not to change the control architecture of PX4 as it yields acceptable results. The designed pitch rate controller yields a well-damped system capable of rejecting disturbance torques.

6.1.5 Angle Controllers

The angle controllers allow the vehicle to follow a given roll, pitch, and yaw angle. PX4 implements a quaternion based angle control law from [44], which is briefly introduced.

The error quaternion is calculated as

$$\bar{\boldsymbol{q}}_e = \bar{\boldsymbol{q}}^{-1} \cdot \bar{\boldsymbol{q}}_r, \tag{6.10}$$

where $\bar{\boldsymbol{q}}$ is the current attitude of the quadrotor, $\bar{\boldsymbol{q}}_r$ is the desired attitude of the quadrotor and $\bar{\boldsymbol{q}}_e$ is the error quaternion, representing the rotation from $\bar{\boldsymbol{q}}$ to $\bar{\boldsymbol{q}}_r$. As mentioned in Section 3.2, the error quaternion consists of a magnitude and a vector component, denoted as

$$\bar{\boldsymbol{q}}_e = \begin{bmatrix} q_{0_e} \\ \boldsymbol{q}_{v_e} \end{bmatrix}, \text{ and} \tag{6.11}$$

$$\boldsymbol{q}_{v_e} = \begin{bmatrix} q_{1_e} \\ q_{2_e} \\ q_{3_e} \end{bmatrix}. \tag{6.12}$$

The proposed control law given by [44], and implemented by PX4, is

$$\Omega_{\mathcal{B}_{i,r}} = 2P_{q_i}\text{sgn}(q_{0_e})q_{j_e} \qquad \text{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{6.13}$$

where $j = \{1, 2, 3\}$ and P_{q_i} is the proportional gain of the controller. The angle controller commands an angular rate setpoint $\Omega_{\mathcal{B}_{i,r}}$, which the angular rate controller follows. The control architecture of the angle controller is shown in Fig. 6.8.

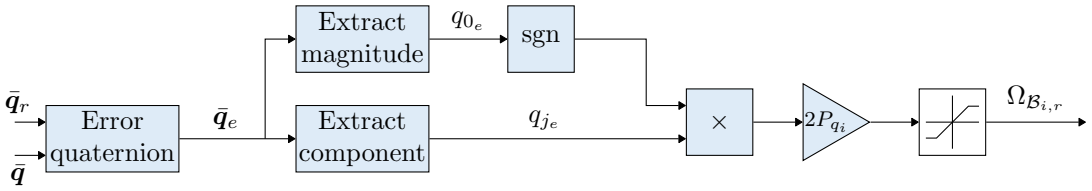


Figure 6.8: The implemented PX4 angle controller.

The angle controller includes a technique that prioritizes the roll and pitch angles over the yaw angle. This is done by applying a weight to the yaw error. Readers can refer to [44] for more detail.

For the controller design, the following assumptions are made to linearise the control law:

- It is assumed that $q_{0_e} > 0$, rendering the signum term to always be 1.
- Small angle approximation is applied to reduce the error quaternion value to $q_{j_e} = q_{j_r} - q_j$.

The resulting linearised control law is a standard proportional gain on the error term, given as

$$\omega_{\mathcal{B}_{i,r}} = 2P_{q_j}\left(q_{j_r} - q_j\right). \tag{6.14}$$

The linear plant for the pitch of the derived model in Chapter 3, is given as

$$G_{q_2}(s) = \frac{q_2(s)}{\Omega_{\mathcal{B}_{Y,r}}(s)} = G^{\text{PX4}}_{\Omega_{Y,cl}}(s) \cdot \frac{0.5}{s}, \tag{6.15}$$

where

$$G^{\text{PX4}}_{\Omega_{Y,cl}}(s) = \frac{\Omega_{\mathcal{B}_Y}(s)}{\Omega_{\mathcal{B}_{Y,r}}(s)} = \frac{D_{\Omega_Y}(s)\,G^{\text{PX4}}_{\Omega_Y}(s)}{1 + D_{\Omega_Y}(s)\,G^{\text{PX4}}_{\Omega_Y}(s)} \tag{6.16}$$

is the closed-loop pitch rate dynamics. Incorporating the effect of the linearised control law, yields the plant

$$G^{\text{PX4}}_{q_2}(s) = \frac{q_2(s)}{\Omega^{\text{PX4}}_{\mathcal{B}_{Y,r}}(s)} = G^{\text{PX4}}_{\Omega_{Y,cl}}(s) \cdot \frac{1}{s}. \tag{6.17}$$

A block diagram of the linear pitch controller and the plant is shown in Fig. 6.9.

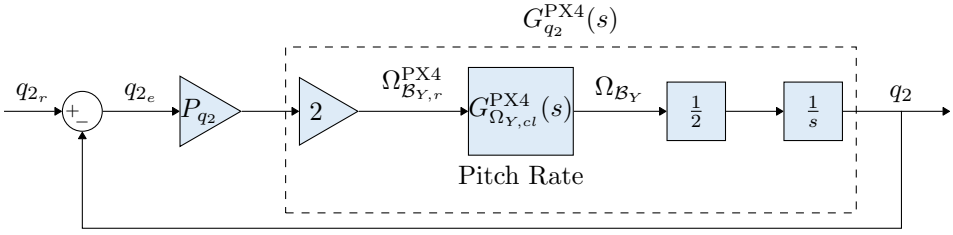


Figure 6.9: Pitch angle controller design block diagram.

The root locus plot of the plant is shown in Fig. 6.10 and the root locus plot of the combined plant and proposed pitch controller is shown in Fig. 6.11. The pitch controller is designed to have an overdamped response that is fast, but still maintains a good time-scale separation to the angular rate controller. The Proportional term of the controller is designed to meet these requirements. An integrator term is not needed, because the surrounding linear velocity and inner angular rate controller both contain integrator terms. The resulting system's most dominant pole is located at $s = -4.64$ on the real axis, resulting in an overdamped system. The bandwidth of the system is 4.41 rad/s, which is 2.84 times smaller than the 12.52 rad/s bandwidth of the angular rate controller, yielding a good time-scale separation.

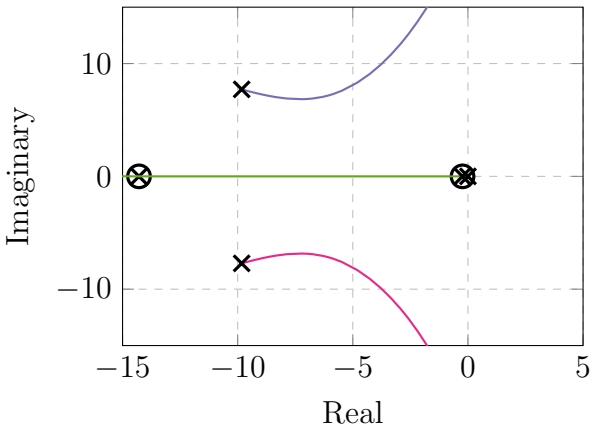


Figure 6.10: Root locus of the pitch dynamics.

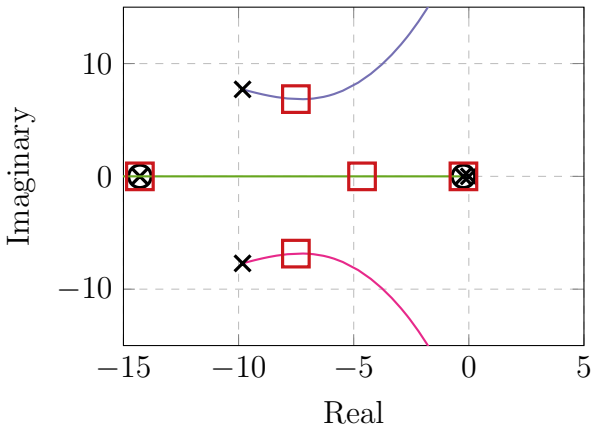


Figure 6.11: Root locus of the pitch controller.

A step response of the pitch controller is shown in Fig. 6.12. The response has no overshoot and a 2% settling time of 0.95 s.

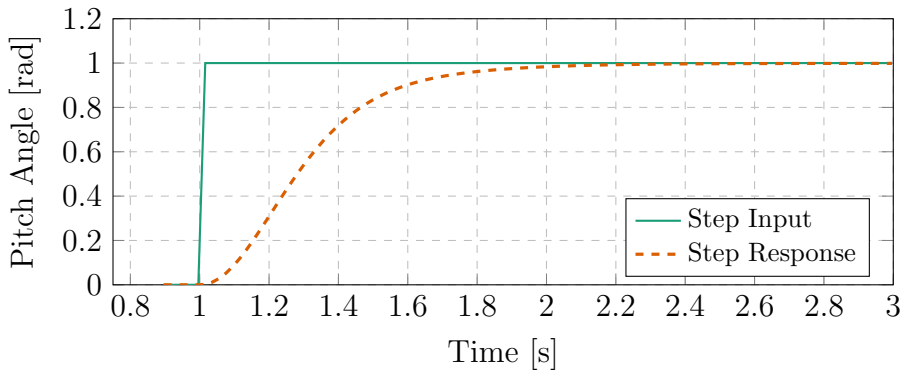


Figure 6.12: Step response of the pitch controller.

6.1.6 Force and Yaw to Attitude and Thrust Conversion

The inertial linear velocity controllers command a force in the inertial frame, driving the vehicle in a specific direction with the desired velocity. This force is transformed to a desired attitude and thrust. The desired attitude is achieved by the inner attitude controllers and the actuators are commanded to achieve the desired thrust.

This transformation is based on work done by [45]. The calculations of the transformation are briefly described in Appendix B.1. The transformation produces a rotation matrix denoting the desired orientation of the vehicle. The desired quaternion is calculated from this rotation matrix and passed to the angle controllers.

6.1.7 Linear Velocity Controllers

The implemented inertial linear velocity controllers have the same architecture as that of the angular rate controllers. Standard PID control is used and the same improvements listed in Table 6.1 is implemented.

The plant of the inertial longitudinal velocity is derived in Appendix B.2. The result is

$$G_{V_N}(s) = \frac{V_N(s)}{q_{2_r}(s)} = G^{\text{PX4}}_{q_2,cl}(s) \cdot \frac{2g}{s}, \tag{6.18}$$

where the closed-loop pitch dynamics are given by

$$G^{\text{PX4}}_{q_2,cl}(s) = \frac{q_2(s)}{q_{2_r}(s)} = \frac{P_{q_2} G^{\text{PX4}}_{q_2}(s)}{1 + P_{q_2} G^{\text{PX4}}_{q_2}(s)}. \tag{6.19}$$

The transformation from the pitch quaternion component to the commanded longitudinal force is given as

$$F_{\mathcal{I}_{N,r}} = 2 m_q g q_{2_r}. \tag{6.20}$$

Therefore, the linear plant describing the dynamics from the commanded longitudinal force to the inertial longitudinal velocity is

$$\frac{V_N(s)}{F_{\mathcal{I}_{N,r}}(s)} = G^{\text{PX4}}_{q_2,cl}(s) \cdot \frac{\frac{1}{m_q}}{s}. \tag{6.21}$$

PX4 makes use of normalized inertial forces. The force $\boldsymbol{F}^{\text{PX4}}_{\mathcal{I}_r}$ is normalized to $[-1, 1]$. The commanded inertial force is related to the normalized force by

$$\boldsymbol{F}_{\mathcal{I}_r} = 4 T_{\text{MAX}} \boldsymbol{F}^{\text{PX4}}_{\mathcal{I}_r}. \tag{6.22}$$

Incorporating the normalized force, changes the plant to

$$G^{\text{PX4}}_{V_N}(s) = \frac{V_N(s)}{F^{\text{PX4}}_{\mathcal{I}_{N,r}}(s)} = G^{\text{PX4}}_{q_2,cl}(s) \cdot \frac{\frac{4 T_{\text{MAX}}}{m_q}}{s}. \tag{6.23}$$

A block diagram with the linear longitudinal velocity controller and the linear plant is shown in Fig. 6.13.

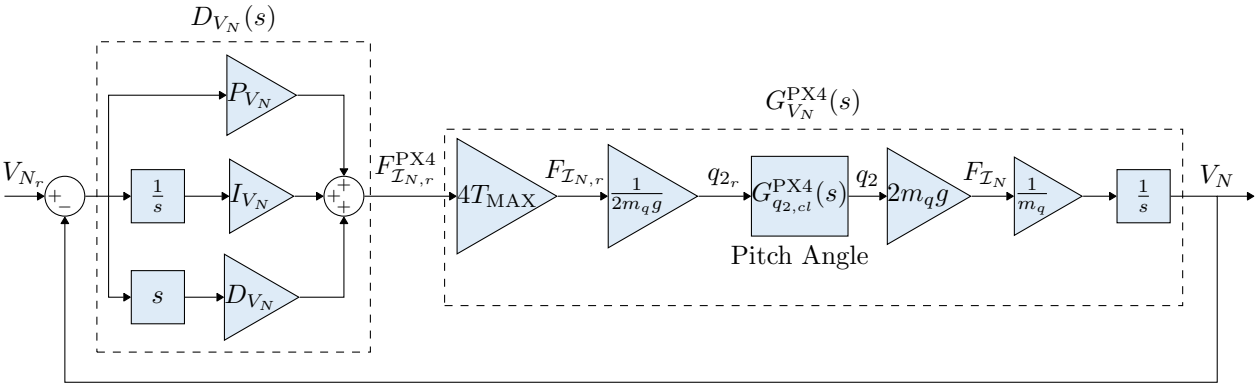


Figure 6.13: Longitudinal velocity controller design block diagram.

The root loci of the plant and the combination of the plant and proposed PID controller are shown in Fig. 6.14 and Fig. 6.15, respectively. The velocity controller is designed with an emphasis on the steady-state tracking performance. The attitude controllers provide stability and are prone to cause the quadrotor to drift due to disturbances. The velocity controller should counter this drift. The same design process is followed as with the angular rate controller.

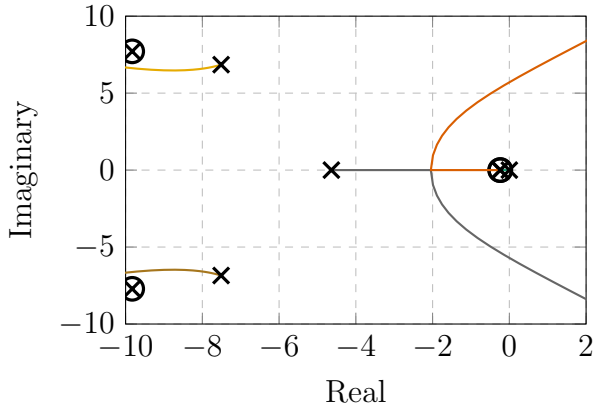


Figure 6.14: Root locus of the longitudinal velocity dynamics.

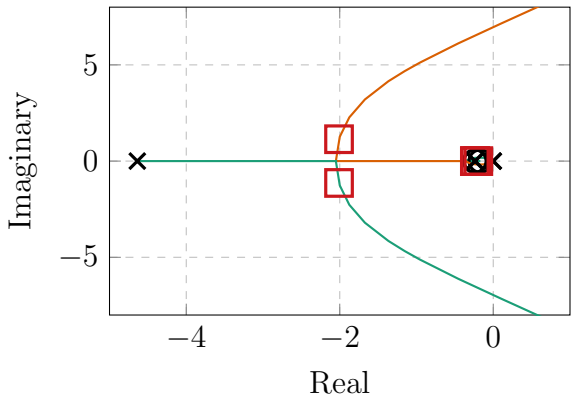


Figure 6.15: Root locus of the longitudinal velocity controller.

The step response of the system is shown in Fig. 6.16. The response has an overshoot of 12%, a 5% settling time of 6.9 s and a 2% settling time of 11.6 s. The same effect is seen as with the angular rate controllers with the long tail due to the integrator term. In this case, a faster transient response is sacrificed for good disturbance rejection. The bandwidth of the longitudinal velocity controller is 2.166 rad/s, which is 2.035 times slower than the pitch angle controller bandwidth of 4.41 rad/s. The performance of the disturbance rejection is seen in Fig. 6.17, which introduces a constant disturbance at time $t = 8$ s.

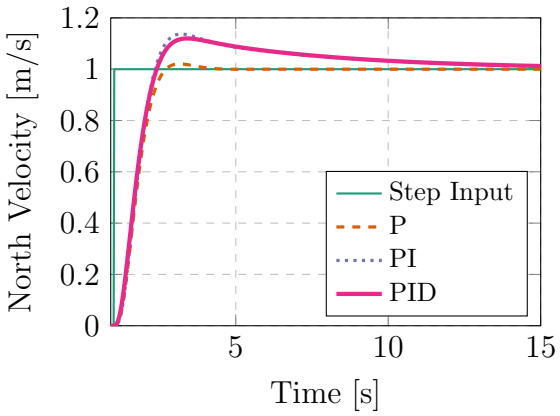


Figure 6.16: Step response of the velocity controller.

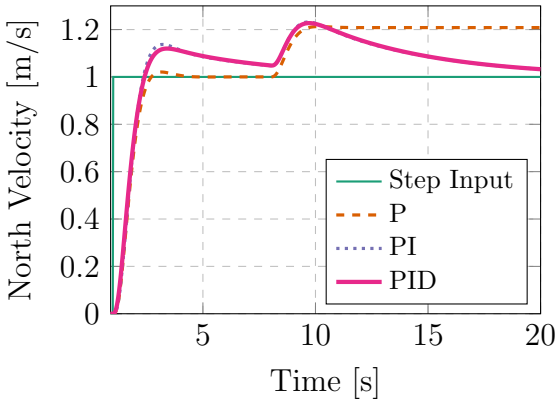


Figure 6.17: Disturbance rejection of the velocity controller.

6.1.8 Position Controllers

The position controllers allow the vehicle to fly to a specific position, specified as NED coordinates. The position controllers are implemented as Proportional controllers and command a

linear velocity $V_{\mathcal{I}_{l,r}}$ to achieve the desired position $X_{\mathcal{I}_{l,r}}$, where $l = \{N, E, D\}$. The control architecture of the position controller is shown in Fig. 6.18. The velocity control signal is limited by an upper and lower bound to limit the flying speed of the vehicle.

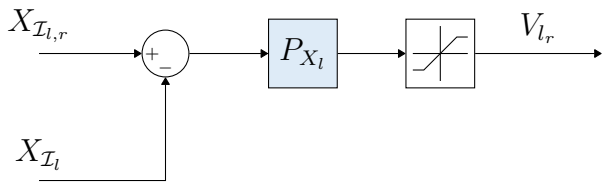


Figure 6.18: The implemented PX4 position controller.

The transfer function describing the closed-loop linear velocity dynamics is given as

$$G^{\text{PX4}}_{V_{N,cl}}(s) = \frac{D_{V_N}(s)G^{\text{PX4}}_{V_N}(s)}{1 + D_{V_N}(s)G^{\text{PX4}}_{V_N}(s)}. \tag{6.24}$$

The linear north position plant is then written as

$$G_{X_N}(s) = G^{\text{PX4}}_{V_{N,cl}}(s) \cdot \frac{1}{s}. \tag{6.25}$$

The block diagram of the linear plant and controller is shown in Fig. 6.19.

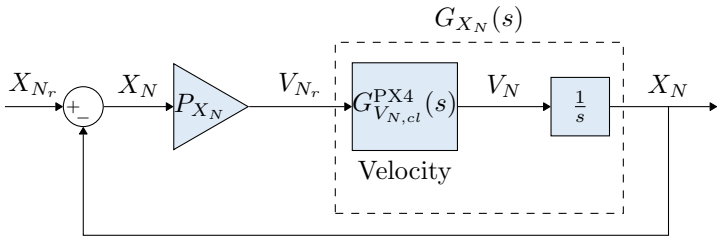


Figure 6.19: North position controller design block diagram.

The design goal of the north position controller is to have a well-damped response. The controller only contains a Proportional term as the inner linear velocity controller has an integrator term for sufficient disturbance rejection. The root locus of the plant is shown in Fig. 6.20 and the root locus of the combined plant and proposed controller is shown in Fig. 6.21.

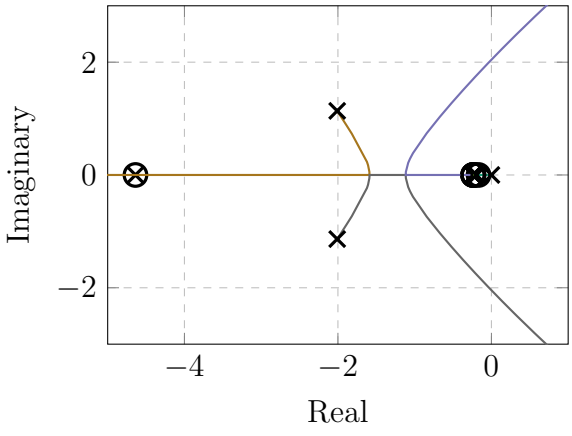


Figure 6.20: Root locus of the north position dynamics.

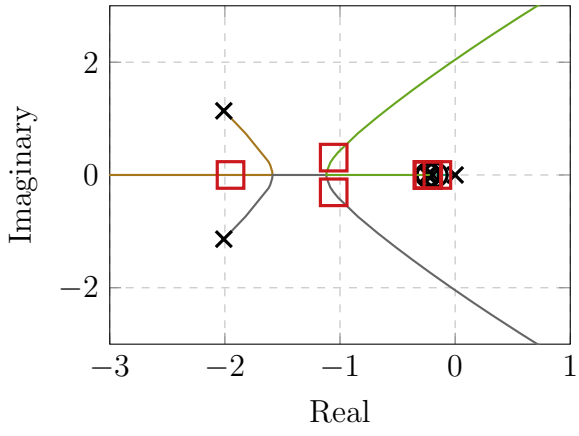


Figure 6.21: Root locus of the north position controller.

A step response of the north position controller is shown in Fig. 6.22. The system response has no overshoot, a 5% settling time of 6.54 s and a 2% settling time of 11.51 s. The bandwidth of the north position controller is 0.59 rad/s, which is 3.69 times slower than the velocity controller bandwidth of 2.166 rad/s.

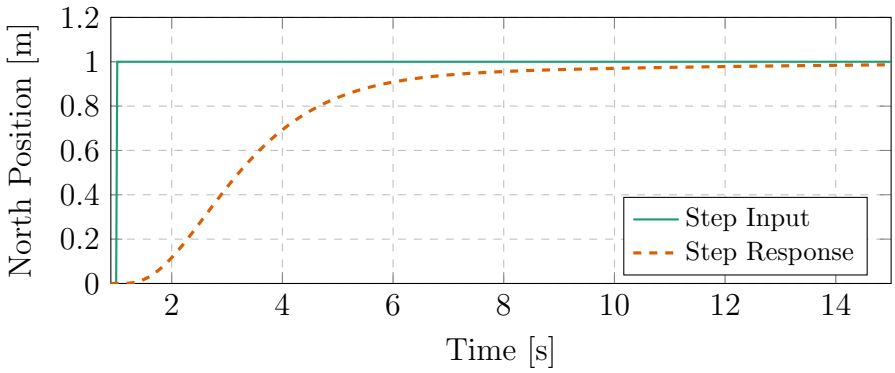


Figure 6.22: Step response of the north position controller.

This concludes the design of the north position controller and the design of the longitudinal controllers. The design of the lateral, heave and directional controllers are discussed in Appendix B.3. The controller gains and parameters are listed in Appendix B.4.

6.2 Simulation Results

The designed controller gains are implemented in the non-linear MATLAB/Simulink and PX4-Gazebo simulation environments. The MATLAB/Simulink simulation environment is used to ensure that the controllers are able to stabilize the quadrotor vehicle. Thereafter, the control gains are implemented in the PX4-Gazebo simulation environment to test the full PX4 stack on the non-linear model.

As mentioned in Chapter 5, these simulation environments simulate the same controllers and non-linear model, but the implementations are vastly different. MATLAB/Simulink is used for fast development and PX4-Gazebo is used to test the full PX4 stack with the added modifications. Therefore, it is important to ensure that the different simulation environments produce the same results. This is illustrated in Figs. 6.23 - 6.26, showcasing the non-linear simulation step responses of the pitch rate, pitch angle, longitudinal velocity and north position controllers.

These results compare the MATLAB/Simulank and PX4-Gazebo responses with the designed linear response. The linear response consists of the linear controller and linear plant from Section 6.1. In order to conduct a fair comparison, no sensor noise or disturbances were included in these simulations. It is clear that the difference between the results is negligible and it is concluded that the implemented controllers satisfy the design requirements and that the MATLAB/Simulink environment closely resembles the PX4-Gazebo simulation environment.

The PX4-Gazebo simulation environment was used to ensure the stability of the vehicle under the influence of high-frequency sensor noise, low-frequency sensor drift, and sensor biases. The result of such a position step response is shown in Figs. 6.27 and 6.28, showcasing the stable flight of the quadrotor.

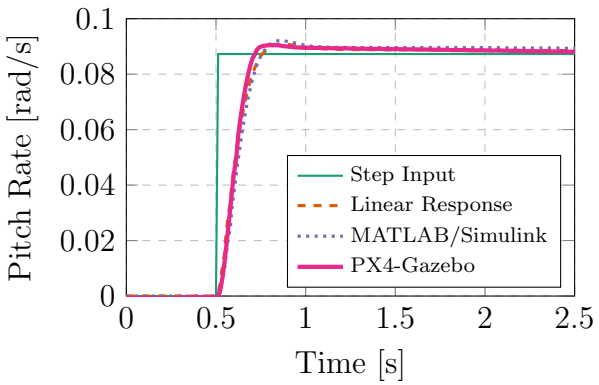


Figure 6.23: Non-linear simulation step response of the pitch rate.

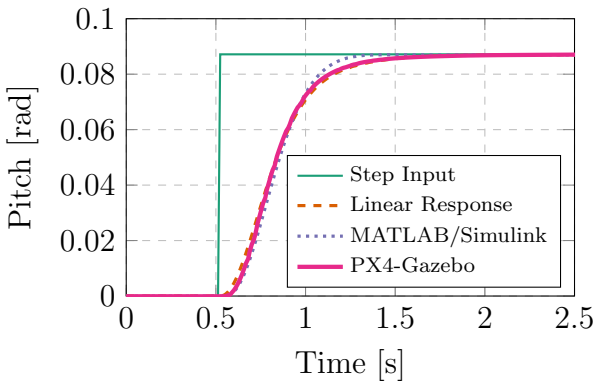


Figure 6.24: Non-linear simulation step response of the pitch angle.

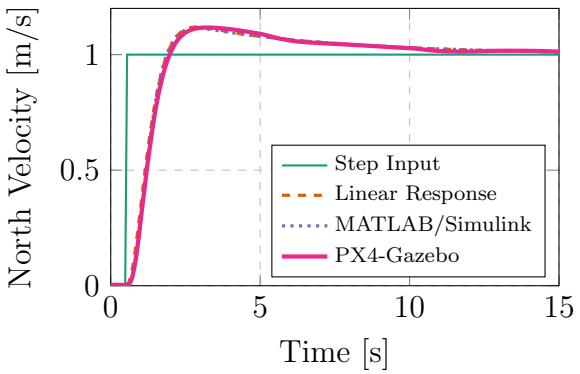


Figure 6.25: Non-linear simulation step response of the longitudinal velocity.

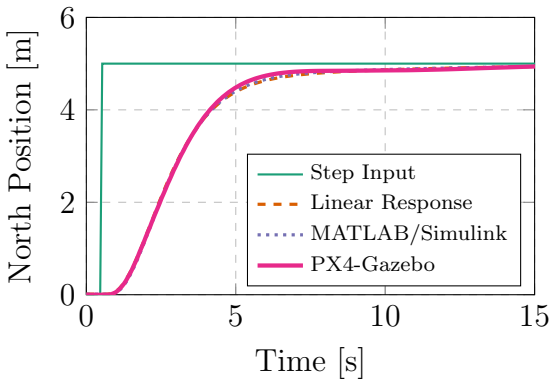


Figure 6.26: Non-linear simulation step response of the north position.

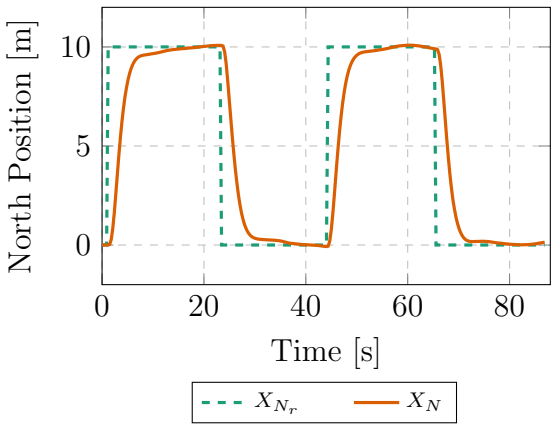


Figure 6.27: The SIL longitudinal position response under the influence of sensor noise.

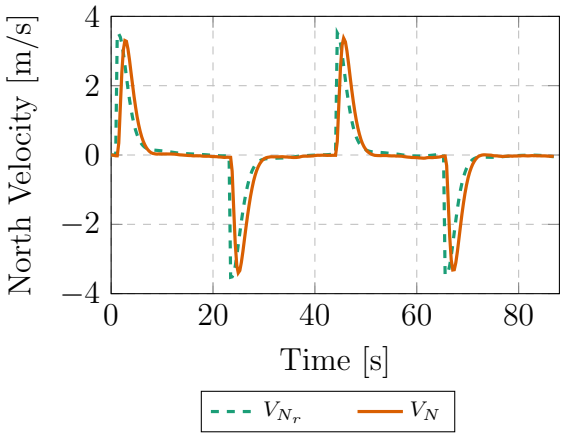


Figure 6.28: The SIL longitudinal velocity response under the influence of sensor noise.

It is clear that the system is still able to meet the design requirements under the influence of sensor noise. Practical flight tests are conducted after the success of the simulation results.

6.3 Practical Flight Test

A series of flight tests were conducted to practically demonstrate the effectiveness of the standard controllers of the quadrotor, shown in Fig. 6.29.

Figure 6.29: Practical flight test quadrotor.

A pilot was involved to manually fly the vehicle in order to progressively test each set of controllers. The flight tests were set out as follows:

1. The pilot flew the vehicle in *Acrobatic (Acro)* mode to ensure the stability of the angular rate controllers.
2. The pilot flew the vehicle in *Stabilized* mode to ensure the stability of the angle controllers.
3. The pilot flew the vehicle in *Position* mode to ensure the stability of the linear velocity controllers.
4. The ROS waypoint scheduler node, introduced in Section 5.4.2, was used to send position commands to the vehicle for autonomous flight.

The outcome of the flight tests were successful. The quadrotor maintained a stable flight during the first 3 manual flight tests and responded correctly to all the inputs that the pilot commanded. The measured motor currents, from the custom motor current module, were all in the same order of magnitude which indicates that the vehicle is in a stable state. The quadrotor experienced quite a lot of position drift during the first 2 flight tests, without any position control. This drift is expected and is a result of effects such as mass inbalance, differences in motors and wind, of which practical systems suffer. The drift was countered during test flights 3 and 4, where the position controllers rejected these disturbances. The position and linear velocity results of the commanded waypoints, using ROS, is shown in Figs. 6.30 and 6.31, respectively.

These results closely resembles the responses seen during simulation. However, the practical position step response has an additional overshoot that was not designed for and does not occur in simulation. This is due to model uncertainties, such as the mass moment of inertia, and the presence of unmodelled disturbances, such as wind or the aerodynamic drag caused by the

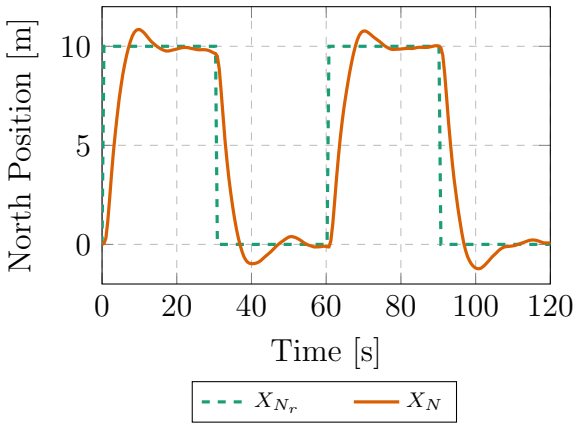


Figure 6.30: Flight test north position response.

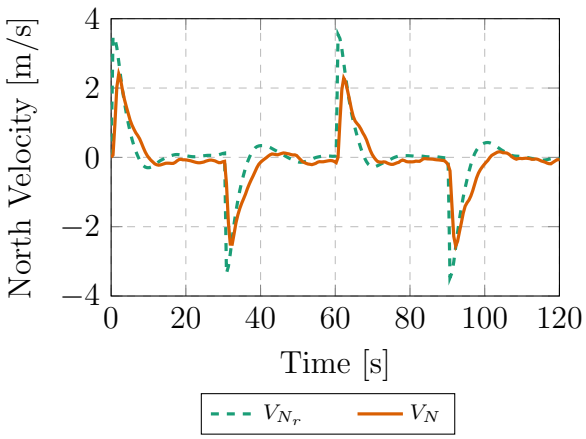


Figure 6.31: Flight test longitudinal velocity response.

propellers. Normally, one would refine and tune the control system when migrating from the simulation environment to the practical system. However, for the purposes of this project, it was decided not to tune these controllers as they will be replaced with modified controllers to account for the effects of the suspended payload on the vehicle. Nonetheless, it is clear from the practical results that the controllers can stabilize the quadrotor during flight.

6.4 Summary

In this chapter, the architecture of the PX4 flight control system was identified, the control gains for the quadrotor vehicle were designed, the quadrotor model and designed controllers were simulated and it practically demonstrated on a physical vehicle. The control system enables autonomous flight of the vehicle and can maintain stable flight. The practical flight results resemble the simulation results, except for the additional noise and overshoot of the position response. Therefore, the simulation environment is a good representation of the physical system. This ensures that the simulation environment is reliable for control design.

The chapter concludes that the control system and physical vehicle is in a working flying condition. Therefore, modifications can now be made to handle the case of carrying an unknown suspended payload.

7. Vehicle with Payload Control System

The focus of this chapter is to design a control system for a multirotor vehicle carrying an unknown suspended payload. The suspended payload significantly alters the flight dynamics of the vehicle as it induces oscillations into the system. Recall the assumption stating that the suspended payload is attached to the CoM of the vehicle. Therefore, the payload will only affect the translational motion of the vehicle and not the rotational motion. The effect of a 2 kg payload with a 1 m rod is seen in the longtudinal velocity response, given a position step input, shown in Fig. 7.1.

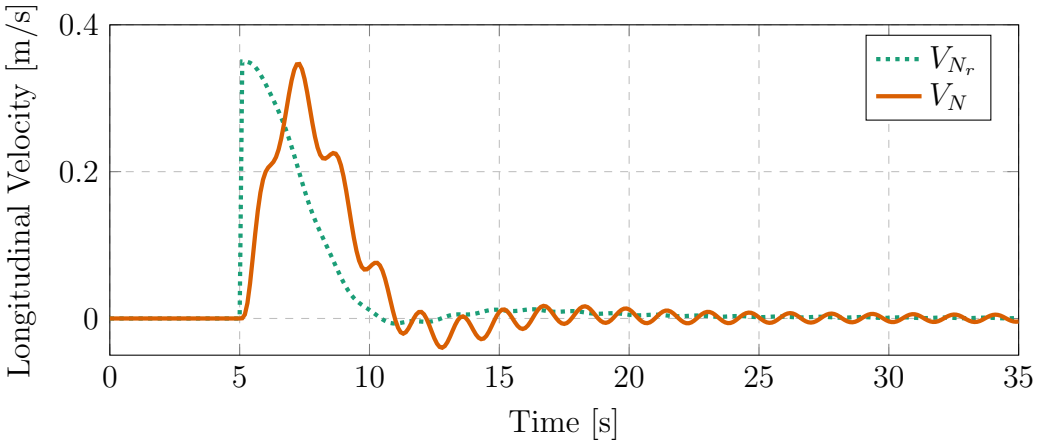


Figure 7.1: The longitudinal velocity of the quadrotor after a position step input.

The control system should be able to damp these oscillations while maintaining stable flight. To achieve this, two approaches are considered for this project, based on the trends seen in Chapter 2. The approaches are:

1. Estimate the payload parameters and the swing angle for use in a full-state feedback controller to simultaneously control the longitudinal velocity of the vehicle and the swing angle of the payload.
2. Implement an adaptive controller able to adapt to the specific attached payload and damp the possible oscillations.

Both of these approaches are explored and simulated in the MATLAB/Simulink environment. Thereafter, they are compared to one another and a single approach is chosen to implement in PX4. The solution is then demonstrated in a practical flight test.

When considering the practical flight test, it is desirable to use a standard PID controller as a baseline to compare it to the results obtained with the other control method. However, oscillations such as those shown in Fig. 7.1 will result in dangerous flight. Therefore, a tuned

PID controller is designed for a specific payload to damp the oscillations and ensure safe stable flight.

This chapter is outlined as follows: Firstly, the relevant plant and equations of motion of a multirotor carrying a suspended payload are given. The design of the tuned PID controller is then presented. Thereafter the estimation and full-state feedback approach is explored, followed by the adaptive control approach. The chapter concludes with a summary and comparison of the two approaches.

7.1 Multitrotor and Suspended Payload Plant

In Chapter 3, the equations of motion of the non-linear model of a quadrotor and suspended payload were derived to simulate the system. In this section, the equations of motion of the system with regards to the controller input and output are introduced. Consider the model shown in Fig. 7.2.

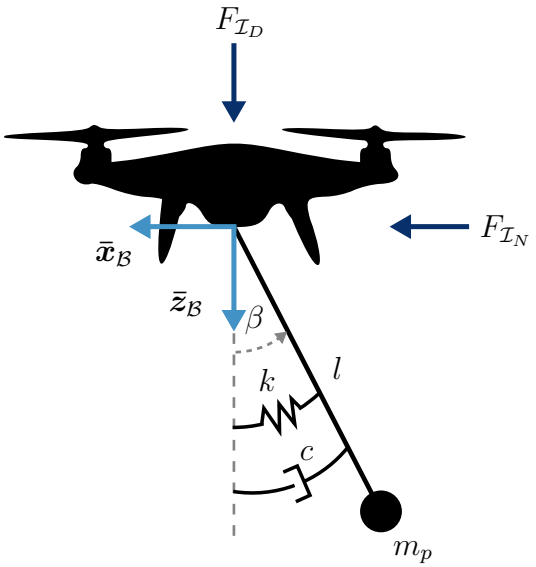


Figure 7.2: The quadrotor and suspended payload system.

The effects of the actuators of the quadrotor are abstracted to the forces $F_{\mathcal{I}_N}$ and $F_{\mathcal{I}_D}$, as these are the inputs to the plant of the linear velocity controllers. The equations of motion of this system are also derived using Lagrangian mechanics, as in Chapter 3. However, the effect of gravity on the vehicle and the effects of the forces $F_{\mathcal{I}_N}$ and $F_{\mathcal{I}_D}$ are now considered. The effect of aerodynamic drag on the payload is omitted. Therefore, the total kinetic and potential energy, respectively, are given as

$$T_e = \frac{1}{2} m_q \left(\dot{x}_q^2 + \dot{z}_q^2\right) + \frac{1}{2} m_p \left(\dot{x}_p^2 + \dot{z}_p^2\right), \text{ and} \tag{7.1}$$

$$V_e = -m_q g z_q - m_p g z_p. \tag{7.2}$$

The set of non-conservative forces are

$$\boldsymbol{Q} = \begin{bmatrix} F_{\mathcal{I}_N} \\ F_{\mathcal{I}_D} \\ -k\beta - c\dot{\beta} \end{bmatrix}. \tag{7.3}$$

The Lagrangian is obtained by $\mathcal{L} = T_e - V_e$ and the Euler-Lagrange equation (3.48) is used to solve the set of differential equations. The equations of motion are obtained as

$$\ddot{x}_q = -\frac{(c\dot{\beta} + k\beta)(m_q + m_p)\cos\beta + l^2 m_q m_p \dot{\beta}^2 \sin\beta}{l m_q (m_q + m_p)} + \frac{m_q + m_p \cos^2\beta}{m_q(m_q + m_p)} F_{\mathcal{I}_N} + \frac{m_p \cos\beta \sin\beta}{m_q(m_q + m_p)} F_{\mathcal{I}_D}, \tag{7.4}$$

$$\ddot{z}_q = \frac{l^2 m_q m_p \dot{\beta}^2 \cos\beta - (c\dot{\beta} + k\beta)(m_q + m_p)\sin\beta}{l m_q (m_q + m_p)} + \frac{m_p \cos\beta \sin\beta}{m_q(m_q + m_p)} F_{\mathcal{I}_N} + \frac{m_q + m_p \sin^2\beta}{m_q(m_q + m_p)} F_{\mathcal{I}_D} + g, \text{ and} \tag{7.5}$$

$$\ddot{\beta} = -\frac{(c\dot{\beta} + k\beta)(m_q + m_p)}{l^2 m_q m_p} + \frac{\cos\beta}{l m_q} F_{\mathcal{I}_N} + \frac{\sin\beta}{l m_q} F_{\mathcal{I}_D}. \tag{7.6}$$

These equations are linearised around hover, $F_{\mathcal{I}_D} = \Delta F_{\mathcal{I}_D} - (m_q + m_p)g$, and using the small-angle approximation. The linearised equations are written in the state-space form as

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{b} F_{\mathcal{I}_N}, \text{ and} \tag{7.7}$$

$$V_N = \mathbf{c}\mathbf{X}, \tag{7.8}$$

where

$$\mathbf{X} = \begin{bmatrix} V_N & \dot{\beta} & \beta \end{bmatrix}^T, \tag{7.9}$$

$$\mathbf{A} = \begin{bmatrix} 0 & -\frac{c}{l m_q} & -\frac{k}{l m_q} - \frac{m_p}{m_q} g \\ 0 & -\frac{c(m_q+m_p)}{l^2 m_q m_p} & -\frac{k(m_q+m_p)}{l^2 m_q m_p} - \frac{m_q+m_p}{m_q} \cdot \frac{g}{l} \\ 0 & 1 & 0 \end{bmatrix}, \tag{7.10}$$

$$\mathbf{b} = \begin{bmatrix} \frac{1}{m_q} & \frac{1}{l m_q} & 0 \end{bmatrix}^T, \text{ and} \tag{7.11}$$

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}. \tag{7.12}$$

From the state-space equations, the plant can be calculated using $G(s) = \mathbf{c}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}$, yielding

$$\frac{V_N(s)}{F_{\mathcal{I}_N}(s)} = \frac{1}{m_q} \cdot \frac{s^2 + \frac{c}{l^2 m_q} s + \frac{k}{l^2 m_q} + \frac{g}{l}}{s\left[s^2 + \frac{c(m_q+m_p)}{l^2 m_q m_p} s + \frac{k(m_q+m_p)}{l^2 m_q m_p} + \frac{m_q+m_p}{m_q} \cdot \frac{g}{l}\right]}. \tag{7.13}$$

The state-space equations and the plant describe the approximate linear dynamics of the system. These equations provide insight to the dynamics of the system and can be used for the controller design. The root locus plot of the plant is shown in Fig. 7.3 with a payload of 2 kg and a rod length of 1 m.

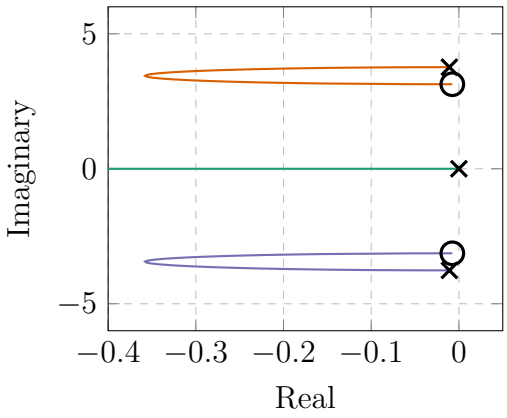
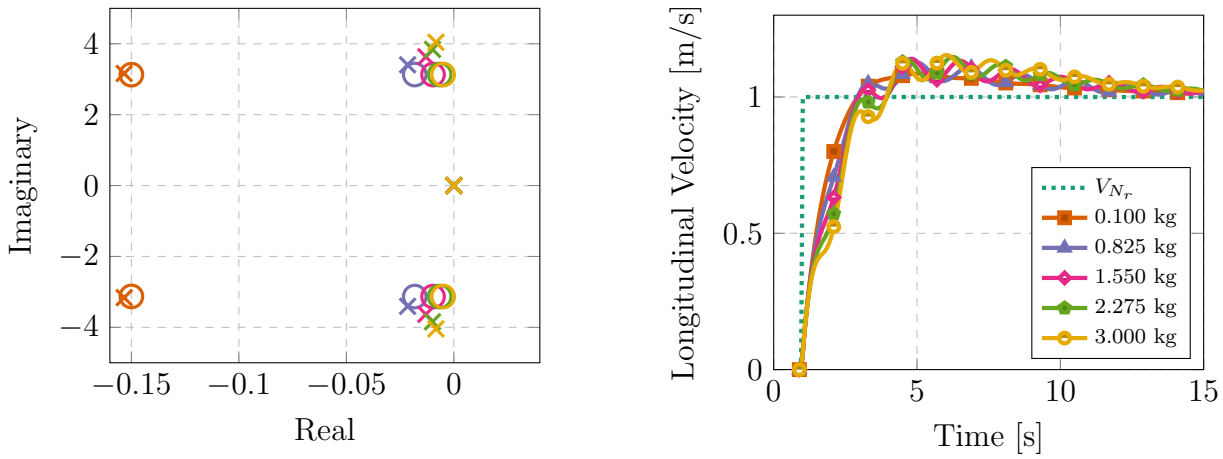


Figure 7.3: Root locus of the longitudinal dynamics of the quadrotor with a suspended payload.

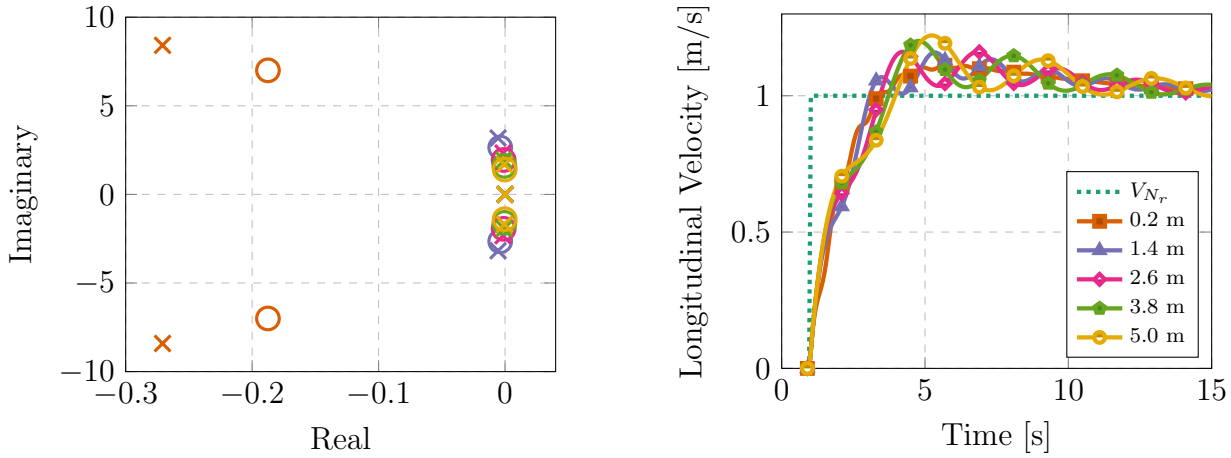
The suspended payload adds two lightly damped poles and zeros to the system. The controller needs to provide enough damping to reduce the effect of these poles and zeros. The parameters of the suspended payload are unknown, and therefore the locations of these poles and zeros are also unknown. This is illustrated in Figs. 7.4 and 7.5, where the step response and root locus plot are shown for payloads of different masses and cable lengths, respectively. The default longitudinal velocity controller designed in Chapter 6 was used to produce these results.



(a) Poles and zeros of the plant.

(b) Step response of the system.

Figure 7.4: The longitudinal velocity dynamics of the quadrotor with suspended payloads of different masses.



(a) Poles and zeros of the plant.

(b) Step response of the system.

Figure 7.5: The longitudinal velocity dynamics of the quadrotor with suspended payloads of different cable lengths.

The root locus plot with different payload masses, see Fig. 7.4a, shows that the real part of the poles and zeros move closer to the origin as the payload mass increases, resulting in a slower response. The poles and zeros also move further away from each other as the mass increases, resulting in less pole-zero cancellation yielding larger oscillations.

The root locus plot with different payload cable lengths, see Fig. 7.5a, shows that the poles and zeros move closer to the origin as the length increases, resulting in a slower response. The poles

and zeros also become less damped, yielding more oscillations in the response as the length increases.

The location of the poles and zeros change with different payloads. Therefore, the control strategy needs to adapt, depending on the payload, to provide sufficient damping for the specific payload.

Two approaches are discussed in future sections to damp the oscillations caused by the payload. The first approach estimates the payload parameters and state in order to design a controller online. The second approach makes use of an adaptive controller to damp the oscillations caused by the unknown payload.

7.2 Tuned PID Controller Design

The tuned PID controller is designed for a payload with known parameters. The goal is to use the tuned PID controller in the practical flight test to provide a baseline result.

A 2 kg payload with a 1 m rod is considered. The design of the tuned longitudinal velocity PID gains follow the same procedure used in Section 6.1.7. The root locus of the plant is shown in Fig. 7.6 and the root locus of the combined plant and controller is shown in Fig. 7.7. The plant is obtained by combining the closed-loop pitch angle controller transfer function, see Eq. (6.19), with the longitudinal velocity transfer function of the vehicle with a suspended payload, see Eq. (7.13).

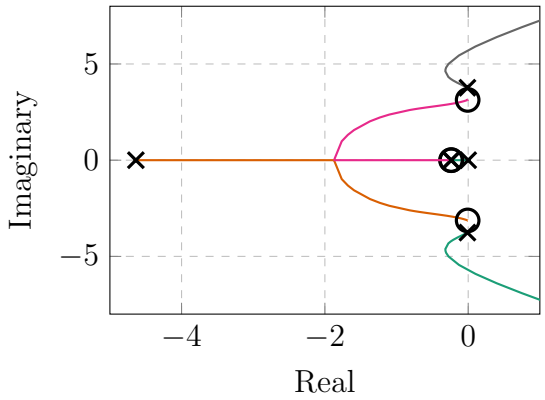


Figure 7.6: Root locus of the velocity dynamics of a quadrotor with a payload.

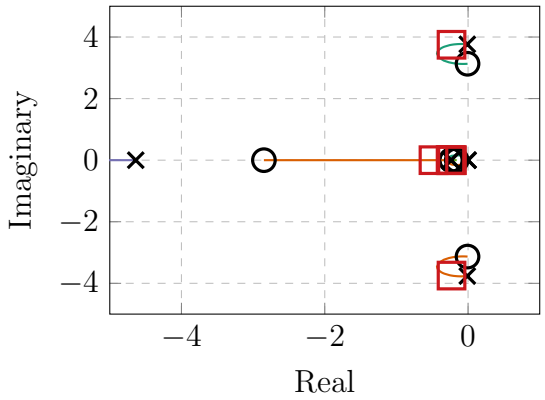


Figure 7.7: Root locus of the tuned PID longitudinal velocity controller.

The pole at $s = 4.4$ rad/s is the dominant pole of the pitch angle controller, which limits the bandwidth of the longitudinal velocity controller. The plant can become unstable and the designed PID controller ensures that the system remains stable and adds more damping into the system to reduce the oscillations caused by the payload. The bandwidth of the designed controller is 0.88 rad/s. The bandwidth may seem slow, but a slower response yields less oscillations and therefore speed was compromised for performance. The step response of the tuned longitudinal velocity controller is shown in Fig. 7.8.

The high overshoot in the step response is due to the integrator term, but it is needed to achieve a zero steady-state tracking error. The oscillations caused by the payload are sufficiently damped to yield a response with an overshoot of 18% and a 5% settling time of 14 s. The tuned

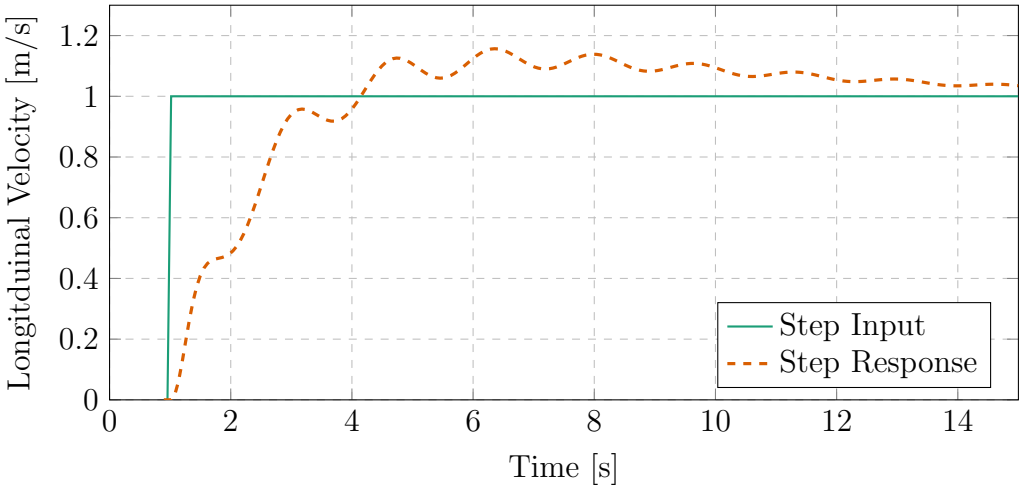


Figure 7.8: Step response of the tuned longitudinal velocity controller.

PID controller yields a response with more damped oscillations than the standard quadrotor controller, designed in Chapter 6. This response is more desirable for the purpose of performing a safe practical flight test. It will serve as a baseline to compare the other control approaches to.

7.3 Estimation and Full-State Feedback Approach

This approach attempts to simultaneously control the longitudinal velocity of the vehicle and the payload swing angle. It is assumed that the swing angle β is not available for measurement, and therefore, an estimate is required to achieve this. From the state-space equations (7.7) - (7.12), it is clear that the swing angle β depends on the payload parameters m_p, l, k, and c. Therefore, the payload parameters need to be estimated first.

The parameters k and c are difficult to estimate. The damping coefficient c models friction at the hinge and the effect of aerodynamic drag. It provides natural damping in addition to the damping that the controller will provide and, therefore, it is not necessary to take into account. The spring coefficient k models the effects of a cable, a flexible rod or the case where the CoM of the payload is far away from the connection point. The parameter k is very small in most cases, and therefore, its effect is considered negligible. The payload mass m_p and cable length l provides the dominant dynamics of the suspended payload and need to be estimated. The payload mass can be obtained by observing the amount of extra thrust required to hover the vehicle with the added mass. The cable length can be obtained by observing the frequency at which the payload oscillates, as the length is proportional to the frequency.

After the estimation of m_p and l, an Linear Quadratic Gaussian (LQG) controller is activated to estimate the swing angle and provide full-state feedback control. An LQG controller consists of an EKF and an LQR controller. The EKF is used to estimate the swing angle β and the LQR controller is a full-state feedback controller responsible for the simultaneous control of the vehicle's longitudinal velocity and the payload swing angle.

A sequence of operation steps are followed to allow the independent estimation of m_p, l and β, after which the LQR controller is activated. It is assumed that the quadrotor's mass is known and that the quadrotor with the suspended payload is already at hover when the estimation

procedure is started. The integrator term of the vertical velocity PID controller allows the quadrotor to hover even with the added unknown payload mass. The sequence of operation steps are:

1. Command an altitude step and estimate m_p using Recursive Least Squares (RLS).
2. Command a position step to induce a small swing angle.
3. Determine the swing frequency from the response.
4. Calculate l from the swing frequency.
5. Start the execution of the EKF to estimate β.
6. Switch from PID to LQR control for the velocity.

After the payload estimation, the LQG controller will control the longitudinal velocity of the vehicle and damp the oscillations caused by the payload.

7.3.1 Payload Mass Estimation

RLS is implemented in the vertical axis to estimate m_p. The inertial vertical dynamics of the system, described in Eq. (7.5), is linearised and the small-angle approximation is applied, yielding

$$\dot{V}_D = \frac{1}{m_q + m_p} F_{\mathcal{I}_D} + g, \tag{7.14}$$

where V_D is the inertial downward velocity. It is desirable to rather use the velocity estimate V_D than the acceleration measurement $\dot{V}_D$, because V_D is estimated by the EKF of PX4, which is more reliable. Therefore, the equation is written in the parametric form

$$z = \boldsymbol{\eta}^T \boldsymbol{\phi} \tag{7.15}$$

where

$$z = \frac{s}{\kappa(s)} V_D, \tag{7.16}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \dfrac{1}{m_q + m_p} & 1 \end{bmatrix}^T, \text{ and} \tag{7.17}$$

$$\boldsymbol{\phi} = \frac{1}{\kappa(s)} \begin{bmatrix} F_{\mathcal{I}_D} & g \end{bmatrix}^T. \tag{7.18}$$

The purpose of the term $\kappa(s)$ is to make the filter $\frac{s}{\kappa(s)}$ proper for implementation purposes, to eliminate the presence of a pure derivative term. The term $\kappa(s)$ is chosen to make the filter $\frac{1}{\kappa(s)}$ at least twice as fast as the dynamics of the actuators. Therefore, it is chosen as $\kappa(s) = s + 30$.

The implemented RLS algorithm is described in [46]. RLS generates estimates of $\boldsymbol{\eta}$ by minimizing the cost function

$$J(\boldsymbol{\eta}) = \frac{1}{2} \int_0^t \left(z(\tau) - \boldsymbol{\eta}^T(t) \boldsymbol{\phi}(t) \right)^2 d\tau \tag{7.19}$$

with respect to $\boldsymbol{\eta}$. The cost $J(\boldsymbol{\eta})$ penalizes all the past errors due to $\boldsymbol{\eta}(t) \neq \boldsymbol{\eta}$. A forgetting factor is included to minimize the effect of possible small payload oscillations during hover.

The RLS equations are given as

$$\frac{d\hat{\boldsymbol{\eta}}(t)}{dt} = \boldsymbol{P}(t)\boldsymbol{\phi}(t)\epsilon(t), \tag{7.20}$$

$$\epsilon(t) = z(t) - \boldsymbol{\phi}^T(t)\hat{\boldsymbol{\eta}}^T(t), \text{ and} \tag{7.21}$$

$$\frac{d\boldsymbol{P}(t)}{dt} = \alpha\boldsymbol{P}(t) - \boldsymbol{P}(t)\boldsymbol{\phi}(t)\boldsymbol{\phi}^T(t)\boldsymbol{P}(t), \tag{7.22}$$

where $\hat{\boldsymbol{\eta}}(t) = [\hat{\eta}_1(t), \hat{\eta}_2(t)]^T$ are the estimates of $\boldsymbol{\eta}$, $\boldsymbol{P}(t)$ is the covariance matrix and α is the forgetting factor. It was found that a forgetting factor of $\alpha = 0.2$ produces good results. The term $\hat{\eta}_2(t) = 1$ is not allowed to change, as the variable g is completely known. The estimated payload mass can now be calculated by

$$\hat{m}_p = \begin{cases} \frac{1-m_q\hat{\eta}_1(t)}{\hat{\eta}_1(t)} & \text{if } \hat{\eta}_1(t) \neq 0 \\ 0 & \text{if } \hat{\eta}_1(t) = 0 \end{cases}. \tag{7.23}$$

The results of the RLS estimation for 1 kg, 2 kg and 3 kg payload masses are shown in Fig. 7.9. The equations are implemented and standard Euler integration is used. The sampling rate is 50 Hz, which is the rate at which the velocity controller runs.

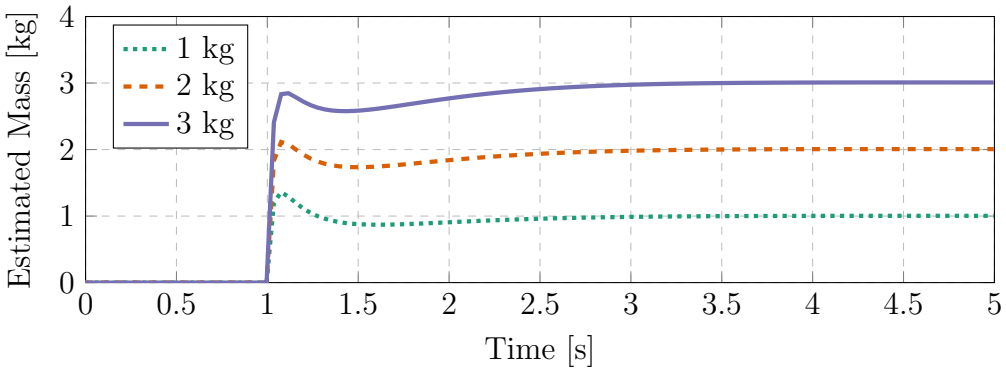


Figure 7.9: The RLS estimates of the payload mass.

It is clear, from the results, that the payload mass is estimated correctly within 3 s. This is the first of the operation steps and a 5 s window is allocated for the payload mass estimate to converge.

7.3.2 Cable Length Estimation

The next step is to calculate the payload cable length. The natural frequency of the quadrotor and suspended payload system is described by [21]

$$\omega_n = \sqrt{\frac{g}{l} \cdot \frac{m_q + m_p}{m_q}}. \tag{7.24}$$

It is clear that the cable length can be calculated from the oscillation frequency. Taking the Fast Fourier Transform (FFT) of the longitudinal velocity, yields the frequencies present in the response signal. The dominant frequency at steady-state will be the oscillations caused by the payload. Therefore, after identifying the dominant frequency from the FFT, the cable length can be calculated. The FFT of the signal in Fig. 7.1 is shown in Fig. 7.10.

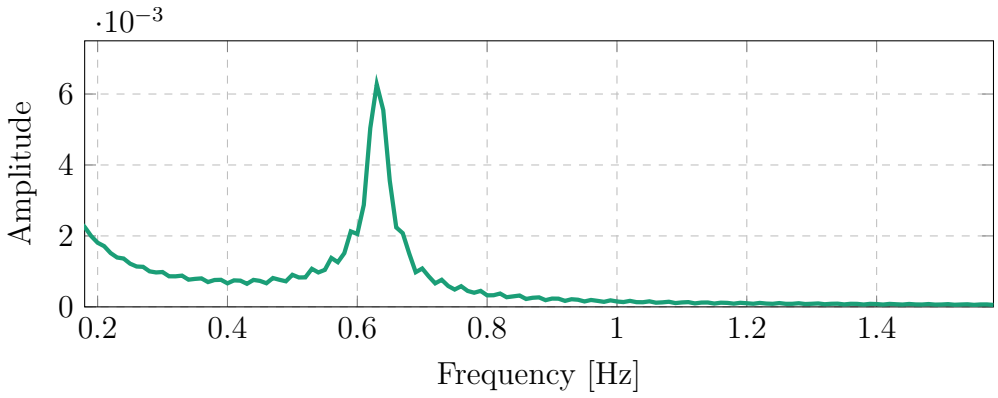


Figure 7.10: The single-sided amplitude spectrum of the north velocity.

The dominant frequency of this signal is 0.63 Hz and by using Eq. 7.24, the cable length is obtained as 0.904 m, which is acceptable as the true length is 1 m. The error in the length estimation may be due to the force $F_{\mathcal{I}_N}$ not having an immediate effect on the system, because of the attitude dynamics.

The dominant frequency will not always have the largest amplitude, because there is a DC component involved in a position step input. In the case where the oscillatory frequency is slow, the peak might be lower than the amplitude of the DC component. To avoid this, the first 5 s of the step response is not included in the FFT and the oscillatory frequency is identified by obtaining the largest peak. A simple algorithm is implemented to obtain this, which identifies all of the peaks in the FFT of the signal and takes the largest one. A peak is identified by considering frequency 3 samples at a time. If the middle sample is larger than its two neighbors, the sample is considered a peak.

Zero-padding is applied to the FFT signal to increase the frequency resolution for better cable length estimates. A resolution of 0.01 Hz is desirable and therefore, the longitudinal velocity signal is zero-padded to 100 s.

A series of simulations were run to test the robustness of the cable length estimator for various payload masses. The results are shown in Fig. 7.11.

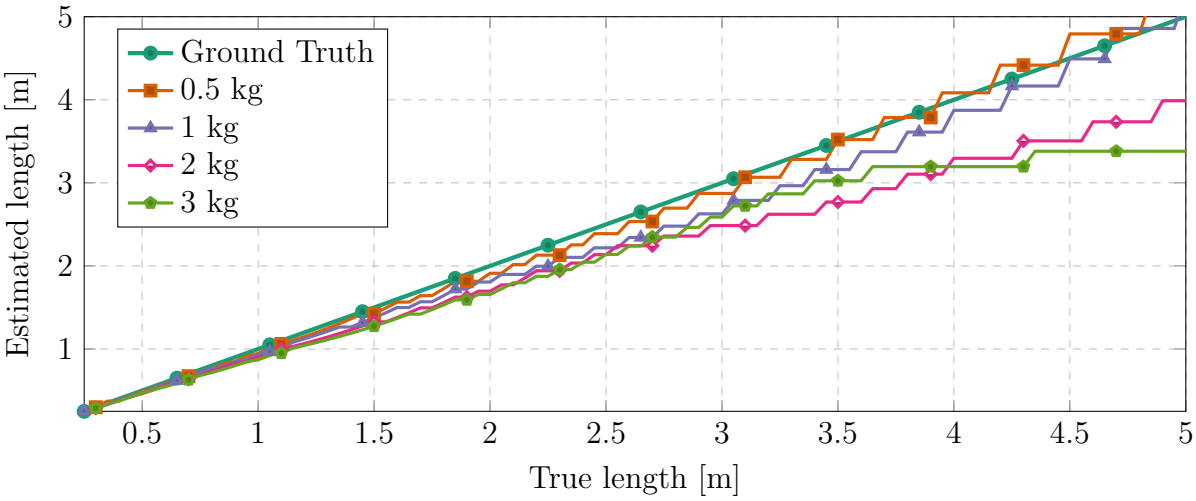


Figure 7.11: The estimated rod length for various payloads.

The following are observed from the results:

- The estimated length has larger errors with heavier payloads, due to the less swing they produce.
- The stair-like result is due to the discrete resolution of the FFT in frequency.

The cable length estimation error increases as the cable length increases with heavier payloads, and the results show a maximum error of 30% for a 5 m link. The effect of such a large error is discussed in the next sections where the consequences are shown in the EKF estimates. However, this is not concerning as such a large payload link will typically not be used in practice. Typically, larger links are used for larger and heavier vehicles as they will not be able to react fast enough to damp the fast oscillations caused by small payload links. Nonetheless, a 5 m link will rarely be required in practice.

The cable length estimation operation step is performed during $5 \leq t \leq 45$ s, after the payload mass estimation. The cable length is, therefore, available at $t = 45$ s for the online design of the LQG controller.

7.3.3 EKF for the Payload Swing Angle Estimation

The EKF component of the LQG controller is used to estimate the payload swing angle β. The payload parameters m_p and l are required for the estimation of β, and therefore, the accuracy of β is dependant on the accuracy of the estimates of m_p and l. A standard EKF is implemented to estimate the state-vector $\mathbf{X}$, given in Eq. (7.7). The model is described by the continuous non-linear state function vector

$$\mathbf{f}(\mathbf{X}_t, \mathbf{u}_t) = \begin{bmatrix} \ddot{x}_q & \ddot{z}_q & \ddot{\beta} \end{bmatrix}^T, \text{ where} \tag{7.25}$$

$$\mathbf{X}_t = \begin{bmatrix} V_N & \dot{\beta} & \beta \end{bmatrix}^T \text{ and} \tag{7.26}$$

$$\mathbf{u}_t = \begin{bmatrix} F_{\mathcal{I}_N} & F_{\mathcal{I}_D} \end{bmatrix}^T. \tag{7.27}$$

The differential equations $\ddot{x}_q$, $\ddot{z}_q$, and $\ddot{\beta}$ are given by Eqs. (7.4) - (7.6), respectively, with $k = 0$ and $c = 0$. The output equation is given by

$$\mathrm{h}(\mathbf{X}_t) = V_N. \tag{7.28}$$

The different steps of the EKF are:

1. Propagate the states by using Euler integration. The propagated states at the current timestep, $\hat{\mathbf{X}}_k^-$, are calculated by using the updated states of the previous timestep, $\hat{\mathbf{X}}_{k-1}^+$. The propagated states are calculated as

$$\hat{\mathbf{X}}_k^- = \hat{\mathbf{X}}_{k-1}^+ + T_s \mathbf{f}(\hat{\mathbf{X}}_{k-1}^+, \mathbf{u}_{k-1}), \tag{7.29}$$

   where T_s is the size of a timestep at which the EKF is executed.

2. Propagate the error covariance matrix. The propagated error covariance matrix at the current timestep, $\mathbf{P}_k^-$, is calculated by using the updated error covariance matrix of the previous timestep, $\mathbf{P}_{k-1}^+$. The calculation is given by

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k, \tag{7.30}$$

where $\mathbf{F}_k$ is the Jacobian matrix, calculated by

$$\mathbf{F}_k = \mathbf{I}_3 + T_s\mathbf{F}_t \text{ and} \tag{7.31}$$

$$\mathbf{F}_t = \left.\frac{\partial \mathbf{f}(\mathbf{X}_t, \mathbf{u}_{(k-1)T_s})}{\partial \mathbf{X}_t}\right|_{\mathbf{X}_t=\hat{\mathbf{x}}_{k-1}^+}. \tag{7.32}$$

The matrix $\mathbf{Q}_k$ is the state covariance matrix and is chosen according to the process noise of the system. The matrix $\mathbf{I}_3$ is the identity matrix and subscript indicates the dimensions of the square matrix. The state covariances of β and $\dot{\beta}$ are chosen larger than that of V_N.

3. The Kalman filter gains are calculated by

$$\mathbf{L}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathrm{R}_k\right)^{-1}, \text{ where} \tag{7.33}$$

$$\mathbf{H}_k = \left.\frac{\partial \mathrm{h}(\mathbf{X}_t)}{\partial \mathbf{X}_t}\right|_{\mathbf{X}_t=\hat{\mathbf{x}}_k^-}, \text{ and} \tag{7.34}$$

R_k is the measurement covariance of V_N, which should be chosen according to the noise of V_N.

4. The state measurement update is done by

$$\hat{\mathbf{X}}_k^+ = \hat{\mathbf{X}}_k^- + \mathbf{L}_k \left[V_{N_k} - \mathrm{h}(\hat{\mathbf{X}}_k^-)\right], \tag{7.35}$$

where the longitudinal velocity measurement is used to correct the states.

5. The error covariance measurement update is done by

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{L}_k\mathbf{H}_k)\,\mathbf{P}_k^-. \tag{7.36}$$

After the payload mass and cable length estimation, the EKF is activated. The estimated payload swing angle is shown in Fig. 7.12 when a north position step of 1 m is commanded for a payload of 2 kg with a 1 m rod.

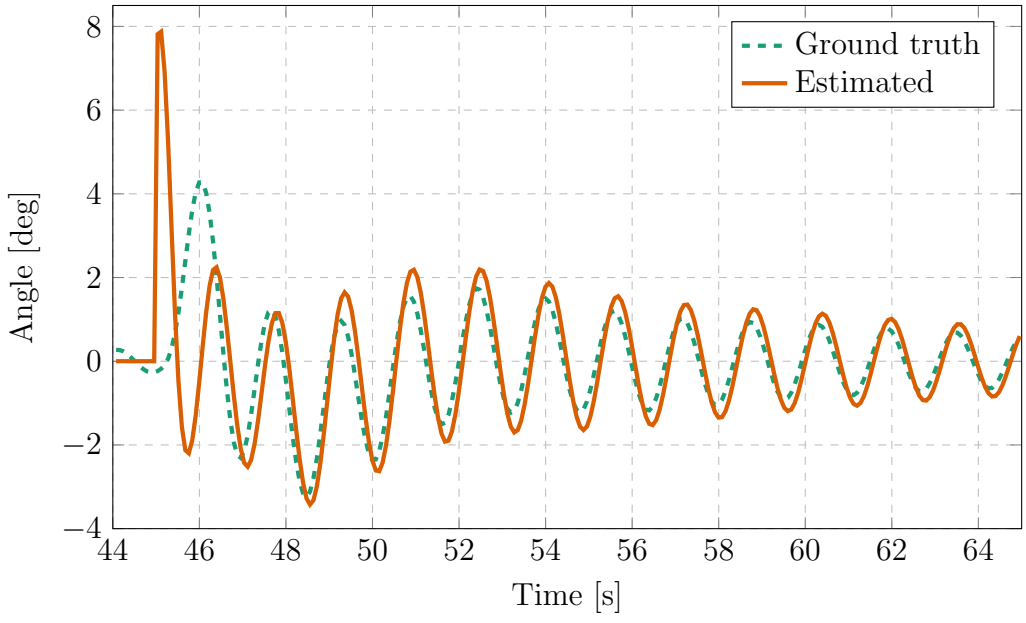


Figure 7.12: The estimated swing angle produced by the EKF.

The estimated angle quickly settles after a transient state and matches the frequency of the ground truth value. However, the following phenomena are observed:

- The estimated angle has a slightly larger magnitude.
- The estimated angle is slightly out of phase.

Both of these phenomena are attributed to the delay in the attitude dynamics, which are not included in the differential equations used in the EKF. The larger magnitude is not considered a problem, as the controller will react more aggressively to reduce the oscillations. The biggest risk of a phase delay is that the swing angle estimate error becomes so large that the vehicle starts amplifying the oscillations. An error in the cable length estimate might lead to such large swing angle estimate errors. This was investigated in simulation and the result of the estimated swing angle for a 3 kg payload with a 5 m rod, which produces a cable length error of 30%, is shown in Fig. 7.13.

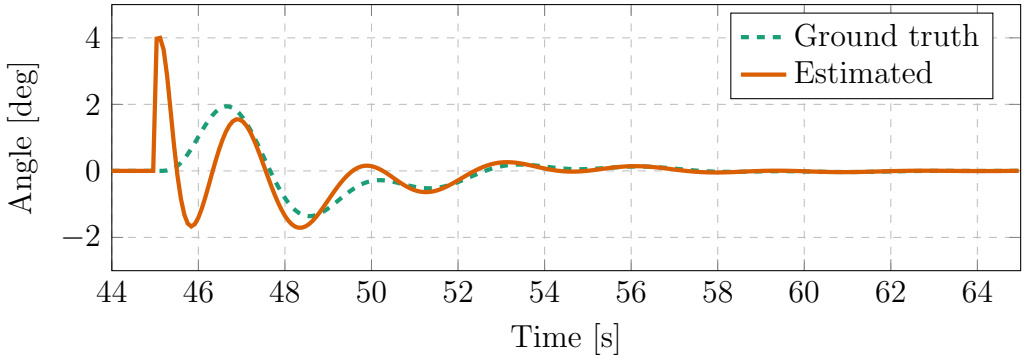


Figure 7.13: The estimated swing angle produced by the EKF with a 30% cable length error.

The EKF is still able to produce a good estimate of the payload swing angle. Therefore, it is concluded that large errors, such as 30%, of the cable length estimate does not have much of an effect on the accuracy of the EKF. The swing angle estimate produced by the EKF is a good representation of the true angle and can be used for control purposes.

7.3.4 LQR Control

The optimal control technique, LQR, is applied to control the system. It is a full-state feedback control technique, which enables the simultaneous control of the longitudinal velocity and payload swing angle. The LQR algorithm calculates the optimal control gains to stabilize the system. However, as it does not contain an integrator, it will not reject disturbances. Therefore, the state-space representation is augmented with an integrator state, given as

$$\dot{V}_{N_I} = V_{N_r} - V_N = V_{N_r} - \mathbf{c}\mathbf{X}, \tag{7.37}$$

which results in the state-space representation

$$\dot{\mathbf{X}}_A = \begin{bmatrix} \dot{V}_{N_I} \\ \dot{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} 0 & -\mathbf{c} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} V_{N_I} \\ \mathbf{X} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} F_{\mathcal{I}_N} + \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} V_{N_r}. \tag{7.38}$$

The control law is given as

$$F_{\mathcal{I}_N} = -\mathbf{K}\mathbf{X}_A, \tag{7.39}$$

where the optimal control gains $\mathbf{K}$ are calculated to minimize the cost function

$$J = \int_0^\infty \left(\mathbf{X}_A^T \mathbf{Q} \mathbf{X}_A + F_{\mathcal{I}_N}^2 \mathrm{R}\right) dt, \tag{7.40}$$

where $\mathbf{Q}$ is the state weighting matrix and R is the input weight. The states β and $\dot{\beta}$ are weighted more than V_N to allow more damping into the system.

After the estimation of the payload mass and cable length, the LQR control gains are calculated. At the same time, the EKF starts executing and the PID controller switches to the full-state feedback LQR controller. The results are shown in Fig. 7.14, with the PID controller active during $0 \leq t \leq 45$ s and the LQR controller active from $t > 45$ s for a payload of 2 kg and a 1 m rod.

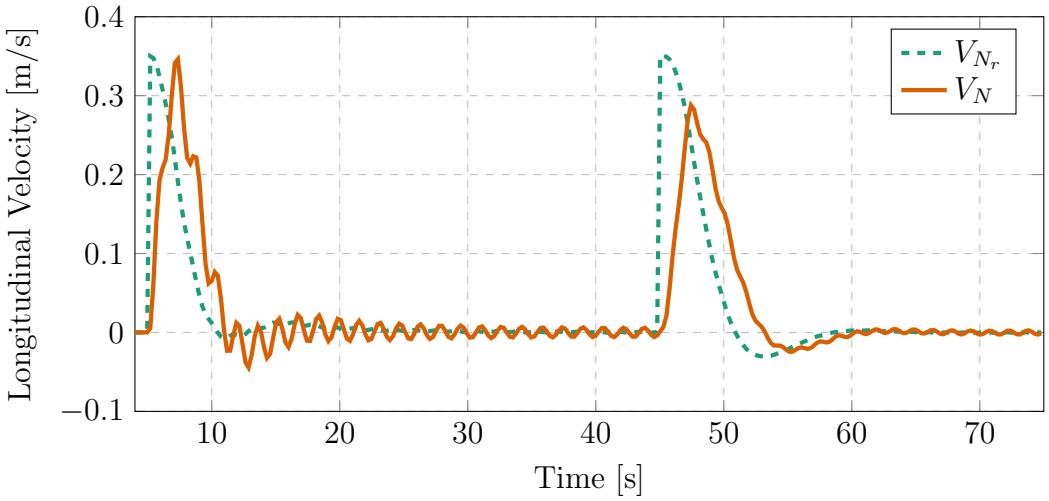


Figure 7.14: The quadrotor's north velocity with PID and LQR control.

The LQR controller significantly reduces the oscillations caused by the payload. The result indicates that this approach is a viable one to introduce more damping into the system. The result is obtained with the ideal case, meaning that there is no sensor noise or disturbances present in the system. Practical systems suffer from these phenomena and, therefore, the robustness of this approach against these effects is investigated.

Robustness

Multirotors often suffer from disturbances due to mass instability, differences in the motors and wind. Therefore, it is important to ensure that the controllers can actively reject disturbances. The LQR controller was augmented with an integrator state to achieve a zero steady-state tracking error in the presence of disturbances. The response of the system, with a constant disturbance introduced at time $t = 65$ s, is shown in Fig. 7.15. The augmented LQR controller can damp the disturbance to achieve a zero steady-state tracking error.

Practical sensors suffer from noise and the effect of noise on the system is explored. The sensor noise is modeled as the sum of high-frequency noise and low-frequency drift, which is added to all of the states of the vehicle. It was found that noise does not greatly affect the accuracy of the payload estimation. The FFT is still able to identify the oscillating frequency as the effect

of noise mainly affects the higher frequency terms. The effect of noise on the RLS algorithm is minimized due to the 5 s window and, therefore, the payload mass estimate remains mainly unaffected. The EKF is known to be quite robust against noise as it is taken into account in the calculations. The response of the system in the presence of noise is shown in Fig. 7.16. The sensor drift causes the vehicle to move around slightly, which induces small oscillations into the system as the payload swings. The LQR controller is able to damp these oscillations and keep them from enlarging. The system seems to be quite robust against sensor noise.

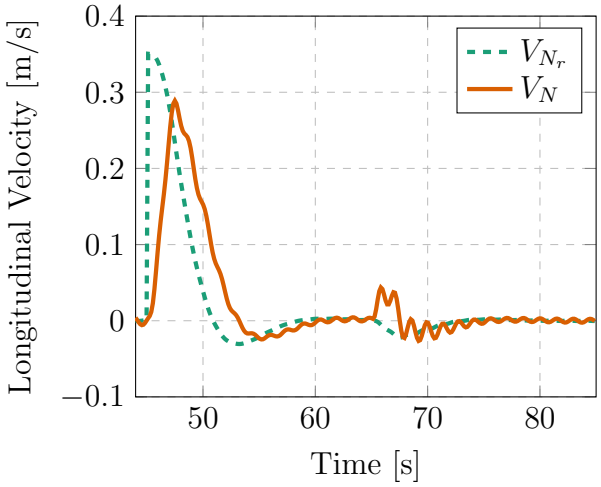


Figure 7.15: The quadrotor's longitudinal velocity with LQR control in the presence of a disturbance.

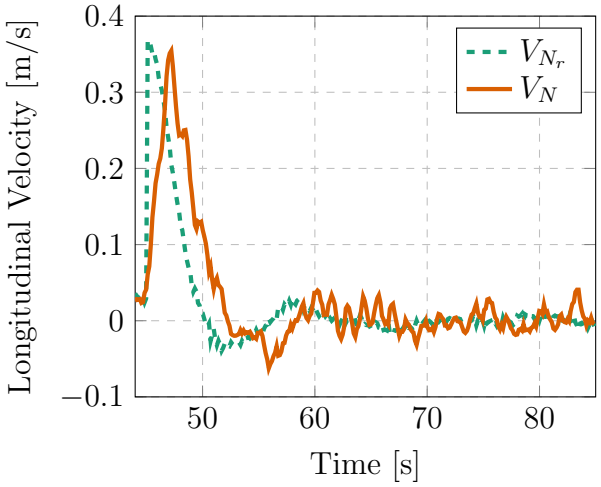


Figure 7.16: The quadrotor's longitudinal velocity with LQR control in the presence of sensor noise.

Flying at high speeds poses interesting results due to the effect of aerodynamic drag. This results in an offset in the payload swing angle when flying at a constant velocity, as shown in Fig. 7.17. It was found that the EKF does not estimate the correct swing angle in this case. However, it estimates an equivalent angle, $\Delta\beta = \beta - \beta_0$, around the new trim angle, β_0, that still describes the effect of the payload on the vehicle. The LQR controller is still able to reduce this effect, which is shown in Figs. 7.18 and 7.19. The LQR response is compared to the standard PID controller response, designed in Chapter 6, which is unaware of the payload. These results are obtained using all of the previous steps.

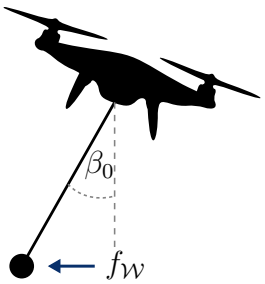


Figure 7.17: The effect of aerodynamic drag on the payload swing angle.

As previously seen, the magnitude of the estimated swing angle is larger than that of the true angle, but is not considered a problem as it only results in a more aggressive controller. The phase error is larger due to the higher velocity, but the results prove that the LQR controller can still damp and reduce the oscillations.

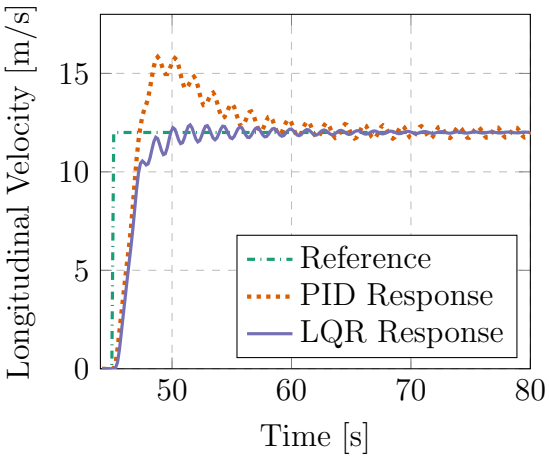


Figure 7.18: Comparison of PID and LQR control with aerodynamic drag.

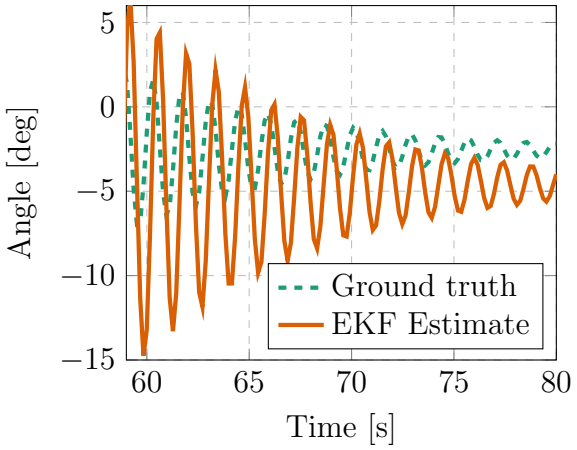


Figure 7.19: Steady-state LQG swing angle estimate with aerodynamic drag.

The true payload swing angle result is offset due to the effect of aerodynamic drag on the payload. The estimated swing angle is also offset, but this is due to the effect of aerodynamic drag on the vehicle. The EKF places the constant disturbance caused by the aerodynamic drag on the vehicle in the angle estimate to compensate for it. The oscillatory effect of the payload on the vehicle is still present in the EKF estimate around the offset. Therefore, the LQR controller is still able to damp the oscillations and achieve a zero steady-state tracking error.

It would be possible to rewrite the EKF to include the attitude of the vehicle and an aerodynamic drag term. Then, the EKF will not place the constant disturbance caused by aerodynamic drag in the swing angle term. This will result in a swing angle estimate around zero. It was decided not to extend the EKF in this way as the response of the controller will remain unchanged and there is no benefit in having the swing angle estimate around zero in this case.

7.3.5 Summary

This approach proves to reduce the effect of an unknown suspended payload on the quadrotor. The approach consists of multiple algorithms including RLS and a FFT for the payload parameter estimation, and an LQG controller to stabilize the system. It proves to be robust against external disturbances and sensor noise, of which practical systems suffer. Therefore, it is a viable solution to the quadrotor with an unknown suspended payload problem.

The position controller was not discussed in this section. For different payloads, this approach will generate different gains for the LQR controller. Subsequently, the dynamics of the velocity controller will change for different payloads. Therefore, it is required to perform an online re-design of the position controller as well. This is not a difficult task as the payload parameters, LQR control gains and a desired bandwidth for the position controller are known. The north position can be included in the LQR algorithm for full-state feedback control. However, it was decided to keep the PID position controller separate to adhere to the control architecture of PX4. Keeping them separate allows both position and velocity command inputs. Therefore, this approach requires the online re-design of both the velocity and position controllers.

7.4 Adaptive Control Approach

An adaptive controller aims to adapt and change its controller to the specific attached payload. The suspended payload induces oscillations into the system and the nature of these oscillations depends on the payload parameters. The adaptive controller will attempt to damp these oscillations while adapting the control gains to cater for the current payload parameters.

The proposed adaptive solution makes use of Model Reference Adaptive Control (MRAC). MRAC updates the controller to change the dynamics of the closed-loop system to that of a predefined reference model [47]. The structure of the MRAC scheme is shown in Fig. 7.20.

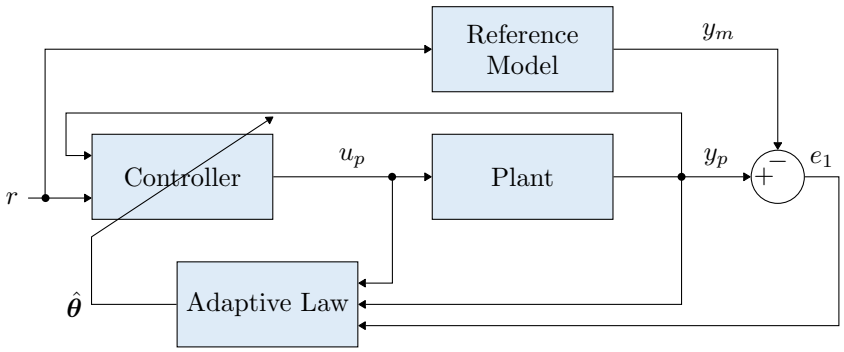


Figure 7.20: Block diagram of the MRAC architecture.

The output of the reference model, y_m, is the desired response of the system given the input r. MRAC produces a control signal u_p by updating the controller parameters with the estimates $\hat{\boldsymbol{\theta}}$ in order to minimize the error between the plant output and the reference model output $e_1 = y_p - y_m$. MRAC effectively performs pole-zero cancellation to transform the closed-loop response to that of the reference model.

Direct MRAC is considered for this project, as opposed to indirect MRAC. The direct case produces estimates of the control parameters, whereas the indirect case produces estimates of the plant after which the control law is updated for these estimates. The direct approach is simpler to implement and the plant parameters are not needed in this case.

The plant and reference model are described by

$$G_p(s) = k_p \frac{Z_p(s)}{R_p(s)}, \text{ and} \tag{7.41}$$

$$W_m(s) = k_m \frac{Z_m(s)}{R_m(s)}. \tag{7.42}$$

The following assumptions are made regarding the plant model:

- $Z_p(s)$ is a monic polynomial of degree m_p, with all its roots on the left half-plane.
- An upper bound n of degree n_p of $R_p(s)$ is known.
- The relative degree $n^* = n_p - m_p$ is known.
- The sign of the high-frequency gain k_p is known.

From the transfer function of the plant given in Eq. (7.13), it is clear that these assumptions are satisfied with $n_p = 3$, $m_p = 2$, and $n^* = 1$. The following assumptions are made regarding the reference model:

- $Z_m(s)$ is a monic polynomial of degree q_m, with all its roots in the left half-plane.
- $R_m(s)$ is a monic polynomial of degree p_m where $p_m \leq n$, with all its roots in the left half-plane.
- The relative degree $n_m^* = p_m - q_m$ of the reference model is the same as that of the plant, i.e. $n_m^* = n^*$.

where $Z_p(s)$ and $R_p(s)$ are polynomials of degree m_p and n_p, respectively, and the relative degree is described by $n^* = n_p - m_p$.

The goal of the adaptive controller is to damp the oscillations caused by the payload. Therefore, a first-order reference model is chosen as it has no oscillations. It is given as

$$W_m(s) = \omega_m \cdot \frac{1}{s + \omega_m}, \tag{7.43}$$

where ω_m is the bandwidth of the reference model. The reference model satisfies all of the assumptions with $p_m = 1$, $q_m = 0$, and $n_m^* = n^* = 1$.

The bandwidth of the reference model is designed to provide a good time-scale separation from the pitch controller. It is desirable that the bandwidth of the reference model is at least 5 times slower than that of the pitch controller. The attitude dynamics are not considered during the adaptive controller design and therefore such a large time-scale separation is needed to ensure that the delay of the attitude dynamics does not influence the adaptive controller. Therefore, the bandwidth is chosen as

$$\omega_m = 0.8 \text{ rad/s}, \tag{7.44}$$

which is 5.51 times slower than the pitch angle controller bandwidth of 4.41 rad/s. The bandwidth of the pitch angle controller adds an upper limit to the frequency it can damp and hence it limits the type of payload which is controllable.

The algorithm consists of a control law, capable of stabilizing the system, and an adaptive law, capable of adapting the controller to minimize the error e_1. These laws are designed separately in the following sections.

7.4.1 Control Law

The MRAC control law is given by [47] as

$$u_p = \boldsymbol{\theta}_1^T \frac{\boldsymbol{\alpha}(s)}{\Lambda(s)} u_p + \boldsymbol{\theta}_2^T \frac{\boldsymbol{\alpha}(s)}{\Lambda(s)} y_p + \theta_3 y_p + c_0 r, \tag{7.45}$$

where

$$\boldsymbol{\alpha}(s) = \begin{cases} \boldsymbol{\alpha}_{n-2}(s) = [s^{n-2} \quad \dots \quad s \quad 1]^T, & \text{for } n \geq 2 \\ 0, & \text{for } n = 1 \end{cases} \tag{7.46}$$

The term $\Lambda(s)$ is an arbitrary monic polynomial, with all its roots in the left half-plane, of degree $n - 1$ that contains $Z_m(s)$ as a factor, given as

$$\Lambda(s) = \Lambda_0(s) Z_m(s). \tag{7.47}$$

The term $\frac{\boldsymbol{\alpha}(s)}{\Lambda(s)}$ serves as a filter and is required for higher order systems to effectively perform pole-zero cancellation to change closed-loop response to that of the reference model. To achieve this, $\Lambda(s)$ is chosen as

$$\Lambda(s) = s^2 + 2\zeta_\lambda \omega_\lambda s + \omega_\lambda^2, \tag{7.48}$$

with the parameters ζ_λ and ω_λ to be designed. The term $\boldsymbol{\alpha}(s)$ is obtained as

$$\boldsymbol{\alpha}(s) = \begin{bmatrix} s & 1 \end{bmatrix}^T. \tag{7.49}$$

A general feedback block diagram of the control law is shown in Fig. 7.21, where

$$C(s) = \frac{c_0 \Lambda(s)}{\Lambda(s) - \boldsymbol{\theta}_1^T \boldsymbol{\alpha}(s)}, \text{ and} \tag{7.50}$$

$$F(s) = -\frac{\boldsymbol{\theta}_2^T \boldsymbol{\alpha}(s) + \theta_3 \Lambda(s)}{c_0 \Lambda(s)}. \tag{7.51}$$

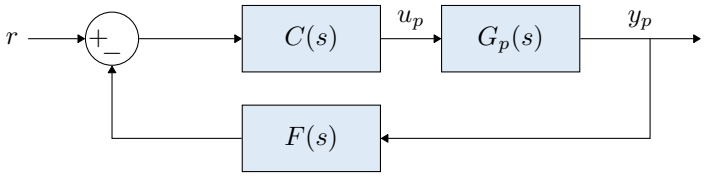


Figure 7.21: General feedback block diagram of the MRAC control law.

In the non-adaptive case, the controller parameter vector $\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}_1^T & \boldsymbol{\theta}_2^T & \theta_3 & c_0 \end{bmatrix}^T$ is chosen in such a way that the transfer function from r to y_p is $W_m(s)$. In the adaptive case, these parameters are replaced with their estimates $\hat{\boldsymbol{\theta}}$. The ideal control parameters $\boldsymbol{\theta}$ are derived by equating the closed-loop system to that of the reference model

$$\frac{C(s)G_p(s)}{1 + C(s)F(s)G_p(s)} = W_m(s). \tag{7.52}$$

The ideal control parameters $\boldsymbol{\theta}$ are then obtained as

$$\boldsymbol{\theta}_1 = \begin{bmatrix} \theta_{11} \\ \theta_{12} \end{bmatrix} = \begin{bmatrix} 2\zeta_\lambda \omega_\lambda - \frac{c}{l^2 m_p} \\ \omega_\lambda^2 - \frac{g}{l} - \frac{k}{l^2 m_p} \end{bmatrix}, \tag{7.53}$$

$$\boldsymbol{\theta}_2 = \begin{bmatrix} \theta_{21} \\ \theta_{22} \end{bmatrix} = \begin{bmatrix} m_q \omega_\lambda^2 \left(4\zeta_\lambda^2 - 1\right) + \frac{m_q + m_p}{l^2 m_p} \left(g l m_p + k - 2c\zeta_\lambda \omega_\lambda\right) \\ 2 m_q \omega_\lambda^3 \zeta_\lambda - \frac{c}{l^2 m_p} \left(m_q + m_p\right) \end{bmatrix}, \tag{7.54}$$

$$\theta_3 = -m_q \left(2\zeta_\lambda \omega_\lambda + \omega_m\right) + \frac{c\left(m_q + m_p\right)}{l^2 m_p}, \text{ and} \tag{7.55}$$

$$c_0 = m_q \omega_m. \tag{7.56}$$

These equations give insight on how to choose the filter parameters ω_λ and ζ_λ, and how to choose the initial values of the control parameters, before adaptation. Notice that if ω_λ and ζ_λ are chosen quite large, it will dominate the control parameters. Therefore, the effect of the payload becomes smaller and the control law becomes more robust against parameter uncertainty. The parameter ζ_λ is chosen as $\zeta_\lambda = 0.9$.

The payload parameters are unknown and, therefore, it is desirable to choose the control parameters $\boldsymbol{\theta}$ without considering the payload. When no payload is assumed, the control parameters $\boldsymbol{\theta}$ reduce to

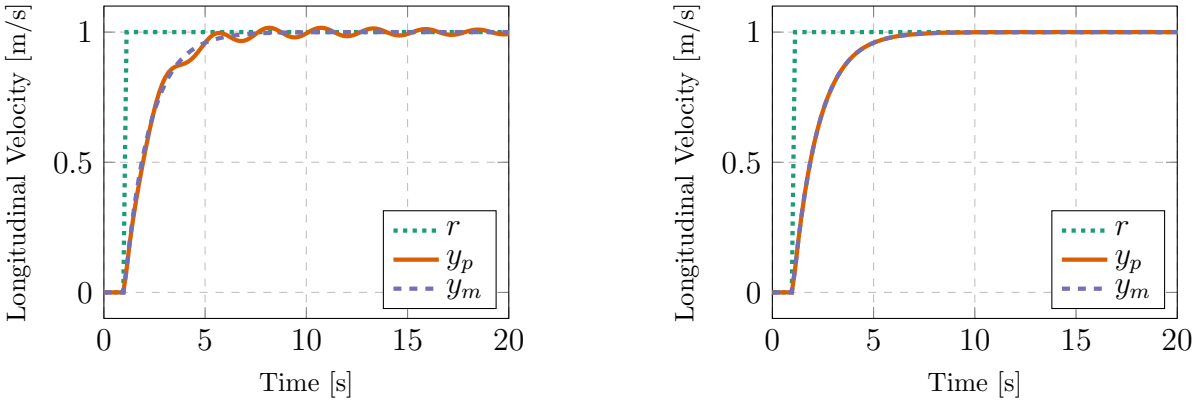
$$\boldsymbol{\theta}_{1_0} = \begin{bmatrix} 2\zeta_\lambda \omega_\lambda \\ \omega_\lambda^2 \end{bmatrix}, \tag{7.57}$$

$$\boldsymbol{\theta}_{2_0} = \begin{bmatrix} m_q \omega_\lambda^2 \left(4\zeta_\lambda^2 - 1\right) \\ 2m_q \omega_\lambda^3 \zeta_\lambda \end{bmatrix}, \tag{7.58}$$

$$\theta_{3_0} = -m_q \left(2\zeta_\lambda \omega_\lambda + \omega_m\right), \text{ and} \tag{7.59}$$

$$c_{0_0} = m_q \omega_m. \tag{7.60}$$

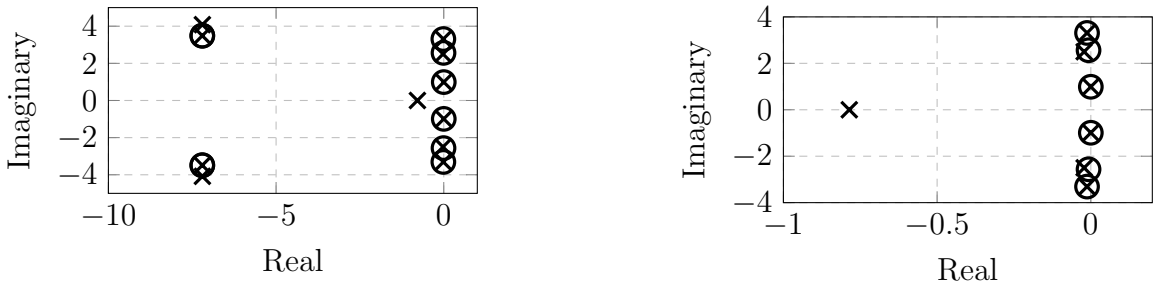
The effect of both small and large values of ω_λ is illustrated in Fig. 7.22, where the MRAC control law is applied to the linear plant for a 3 kg payload with a 1.5 m rod.



(a) $\omega_\lambda = 5$ (b) $\omega_\lambda = 50$

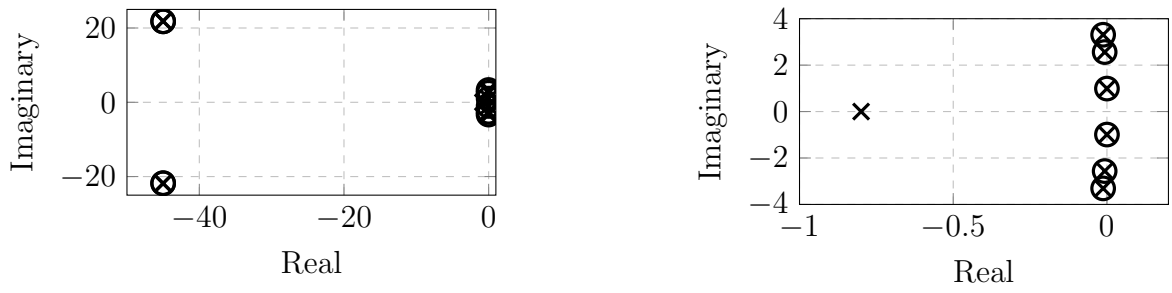
Figure 7.22: The longitudinal velocity response of the linear plant with the MRAC control law.

In the case of $\omega_\lambda = 50$ the response follows the reference model exactly, in contrast to the case of $\omega_\lambda = 5$ where the oscillations caused by the payload are still present. The pole-zero plots of these responses are shown in Figs. 7.23 and 7.24, revealing the effect of the control law on the system. The control law attempts to cancel the poles and zeros related to the dynamics of the payload and moves the pole at the origin to the position relating to the bandwidth of the reference model. The control law also adds poles close to the origin and near the ω_λ location, which the control law attempts to cancel again. The pole-zero plot of the $\omega_\lambda = 50$ rad/s case has better pole-zero cancellation than the $\omega_\lambda = 5$ rad/s case, which explains the improvement shown in the step responses.



(a) All poles and zeros. (b) Poles and zeros near the origin.

Figure 7.23: Pole-zero plot of the closed-loop system with $\omega_\lambda = 5$ rad/s.



(a) All poles and zeros.

(b) Poles and zeros near the origin.

Figure 7.24: Pole-zero plot of the closed-loop system with $\omega_\lambda = 50$ rad/s.

Practically, the parameter ω_λ is considered large if it falls outside of the bandwidth of the velocity dynamics of the vehicle. Therefore, the parameter ω_λ is chosen 10 times larger than the bandwidth of the reference model as

$$\omega_\lambda = 8 \text{ rad/s}. \tag{7.61}$$

From these results, it was proved that the static control law, from Eq. (7.45), can damp the oscillations caused by the payload and that it is robust against parameter uncertainty of the payload, in the linear case. Applying the control law to the non-linear model yields the response shown in Fig. 7.25.

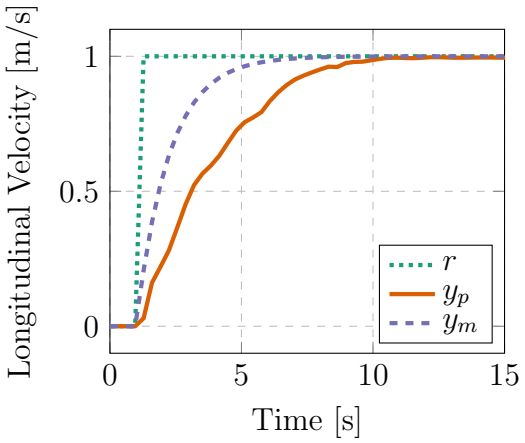


Figure 7.25: The longitudinal velocity response of the non-linear model with the MRAC control law.

The longitudinal velocity response does not sufficiently follow the reference model. Therefore an adaptive law is needed to account for the non-linearities of the system and adjust the control parameters for improved results.

7.4.2 Adaptive Law

An adaptive law changes the control parameters based on the error e_1. The adaptive law is derived using the Strictly Positive Real (SPR)-Lyapunov design approach. This approach involves the derivation of a differential equation that relates the estimation error with the parameter error through a SPR transfer function. Then, a Lyapunov function V is chosen

whose time derivative is made nonpositive, $\dot{V} \leq 0$, by choosing an appropriate adaptive law. The derivation and stability proof of the adaptive law for a general Single-Input-Single-Output (SISO) transfer function can be found in [47]. The final result is obtained as

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = -\boldsymbol{\Gamma} e_1 \boldsymbol{\omega} \text{sgn}\left(\frac{k_p}{k_m}\right), \tag{7.62}$$

where $\boldsymbol{\Gamma}$ is a diagonal matrix containing the adaptive gains of the control parameters, $\text{sgn}(x)$ is the signum function and

$$\boldsymbol{\omega} = \begin{bmatrix} \dfrac{\boldsymbol{\alpha}(s)}{\Lambda(s)} u_p & \dfrac{\boldsymbol{\alpha}(s)}{\Lambda(s)} y_p & y_p & r \end{bmatrix}^T. \tag{7.63}$$

This is known as an unnormalized adaptive law as the amount of adaptation not only depends on the adaptive gains $\boldsymbol{\Gamma}$, but also on the magnitudes of the reference signal r, the output signal y_p, and the control signal u_p. This effect is unwanted in this case, as the velocity commands can vary from small to large values. Therefore, a normalized adaptive law is considered for this project. The normalized adaptive law is given as [47]

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = -\boldsymbol{\Gamma} \epsilon \boldsymbol{\phi} \text{sgn}\left(\frac{k_p}{k_m}\right), \tag{7.64}$$

where

$$\epsilon = \frac{e_1 - \hat{e}_1}{1 + n_s^2}, \tag{7.65}$$

$$\hat{e}_1 = \rho(u_f - \hat{\boldsymbol{\theta}}^T \boldsymbol{\phi}), \tag{7.66}$$

$$\dot{\rho} = \gamma \epsilon (u_f - \hat{\boldsymbol{\theta}}^T \boldsymbol{\phi}), \tag{7.67}$$

$$n_s^2 = \boldsymbol{\phi}^T \boldsymbol{\phi} + u_f^2, \tag{7.68}$$

$$u_f = W_m(s) u_p, \text{ and} \tag{7.69}$$

$$\boldsymbol{\phi} = W_m(s) \boldsymbol{\omega}. \tag{7.70}$$

The normalized adaptive law filters the signals $\boldsymbol{\omega}$ and u_p with the reference model to produce two new signals $\boldsymbol{\phi}$ and u_f. The normalization factor is given by n_s^2, which depends on the magnitudes of $\boldsymbol{\phi}$ and u_f. The normalized adaptive law introduces a new adaptive parameter ρ with an adaptive gain γ. This is used to estimate the output error e_1 and produce a normalized estimation error ϵ. This adaptive law is much more complex than the unnormalized adaptive law. However, the advantage it provides is required to allow the adaptation to remain the same for both small and large velocity commands.

The adaptation of the parameters depends a lot on the type of signals. If the signals are not sufficiently rich in such a way that the dynamics of the system are excited, poor adaptation will take place. Therefore, square-wave learning is used in simulation to test the adaptive controller as a square-wave consists of a very wide range of frequencies which should excite all of the necessary dynamics of the system. Practically, as long as the oscillations caused by the payload are present in the signal, the adaptive parameters will adapt to compensate for the effect. Therefore, the issue of sufficiently rich signals is not a problem for this project.

Appropriate adaptive gains need to be chosen for the adaptive parameters. Given that ω_λ is large, it was decided not to adapt the control parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. This decision is based

on the fact that the poles and zeros related to $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ fall outside of the bandwidth of the longitudinal velocity dynamics and are therefore not excited by the system. Therefore, they require very large adaptive gains for adaptation to take place. Such large gains are undesirable as they can lead to the parameters diverging. This is a side effect of a large ω_λ. However, this is not seen as a problem and the robustness benefit that a large ω_λ gives is more desirable. On the other hand, the parameters θ_3 and c_0 are allowed to adapt to account for the non-linearities in the system and appropriate adaptive gains are chosen. The adaptive gain matrix $\boldsymbol{\Gamma}$ is chosen as

$$\boldsymbol{\Gamma} = \text{diag}\left([0 \quad 0 \quad 0 \quad 0 \quad 2000 \quad 250]\right), \tag{7.71}$$

where the diag($\boldsymbol{x}$) function produces a matrix with $\boldsymbol{x}$ along the diagonal line. These adaptive gains may seem large at first, but because a normalized adaptive law is used, the values are expected to be in this range.

The MRAC response of a quadrotor with a 2 kg payload and 1 m rod length is shown in Fig. 7.26. The change of the adaptive parameters from their initial values is shown in Fig. 7.27.

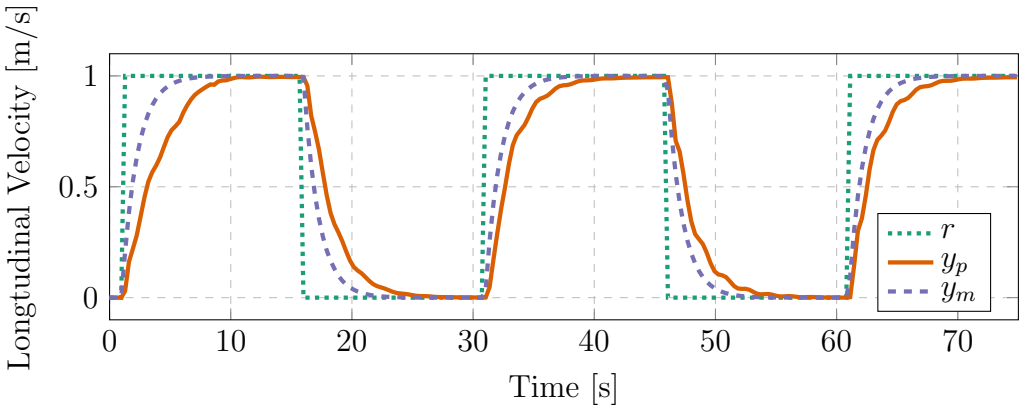


Figure 7.26: The MRAC longtudinal velocity response of the non-linear quadrotor model.

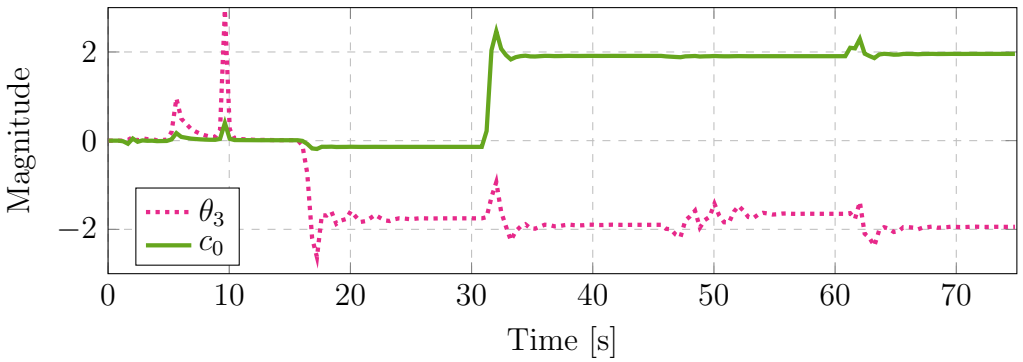
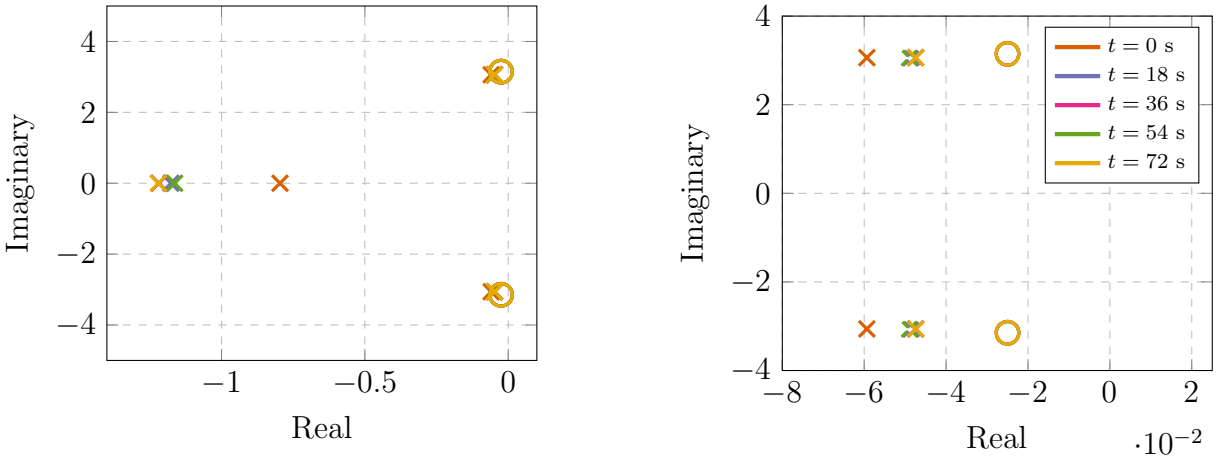


Figure 7.27: The change in the adaptive parameters from their initial values.

The adaptive parameters settle around new values. An improvement is seen in the transient response from the first step response to the subsequent step responses. The response of the system sufficiently follows the reference model. A small error still exists between the plant output and reference model output, because the MRAC algorithm is unable to absorb all of the non-linearities in the system as it is designed for the linear model.

The pole-zero plot of the closed-loop system at different timestamps is shown in Fig. 7.28. Only the dominant poles and zeros that are not effectively canceled are shown in the figure. The plot reveals the change in the closed-loop poles and zeros as the control parameters adapt.



(a) Dominant poles and zeros.

(b) Poles and zeros near the origin.

Figure 7.28: Pole-zero plot of the closed-loop system with changing control parameters at different timestamps.

The pole at $s = -0.8$ is moved further to the left, resulting in a faster response. This is to counter the large transient error present in the first step response. The lightly damped poles near the origin are moved closer to the zeros to cancel their effect and reduce the oscillations caused by the payload.

Fig. 7.26 proves that the MRAC scheme is able to damp the induced oscillations and adapt to the specific payload attached. The adaptive controller is required to work in a practical setup which includes effects such as external disturbances and sensor noise. The effect of these phenomena on the adaptive controller is explored further.

7.4.3 Robustness

Consider the case where $r = 0$ and sensor noise is present in the system. The plant output y_p will not be zero, causing the error e_1 to also not have a value of zero. Considering the adaptive law given in Eq. (7.64), it is clear that the parameter θ_3 will then always adapt in the presence of sensor noise because the adaptation is proportional to the error e_1 and the output y_p. This effect is undesirable as it can lead to the parameter diverging.

Sensor noise includes both high-frequency noise and low-frequency drift. To reduce the effect of high-frequency noise on the adaptive controller, a technique known as a dead-zone can be implemented. The dead-zone technique does not adapt the control parameters when the error e_1 is within certain bounds. These bounds can be chosen according to the covariance of the sensor noise. In this case, the state influenced by noise is the longitudinal velocity. This state will be obtained after it has been estimated by the EKF of PX4. Therefore, the effect of high-frequency noise is negligible. However, low-frequency drift will still be present on the signal, but the dead-zone technique will not be effective in this case.

Another concern is the effect of external disturbances on the system. An external disturbance

causes an offset between y_p and r, and consequently results in an error e_1. This constant error will cause continuous adaptation of the parameters and can lead to them diverging.

A technique known as leakage will be implemented to prevent the parameters from continually adapting in the presence of low-frequency drift and external disturbances. Leakage changes the adaptive law to

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = -\boldsymbol{\Gamma}\text{sgn}\left(\frac{k_p}{k_m}\right)\left[\epsilon\boldsymbol{\phi} + w\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0\right)\right], \tag{7.72}$$

where w is known as the leakage term. Leakage pushes the adaptive parameters back to their initial values, keeping them from diverging. Eventually, both the adaptive and leakage terms reach an equilibrium, stabilizing the parameters. This technique, therefore, trades performance for robustness. Given that the control law is designed to be robust, not much performance will be sacrificed as the system is stable with the initial parameter values.

Multiple options exist for the choice of w. The switching-σ technique is used, which only activates the leakage term when the adaptive parameters are outside some acceptable bounds. The term w is described by

$$w(t) = \begin{cases} 0, & \text{if } |\hat{\theta}| < M_0 \\ \sigma_0\left(\frac{\hat{\theta}}{M_0} - 1\right), & \text{if } M_0 \leq |\hat{\theta}| \leq 2M_0 \\ \sigma_0, & \text{if } |\hat{\theta}| > 2M_0 \end{cases} \tag{7.73}$$

which produces a continuous function for $w(t)$, shown in Fig. 7.29. The variables σ_0 and M_0 need to be designed for each adaptive parameter θ.

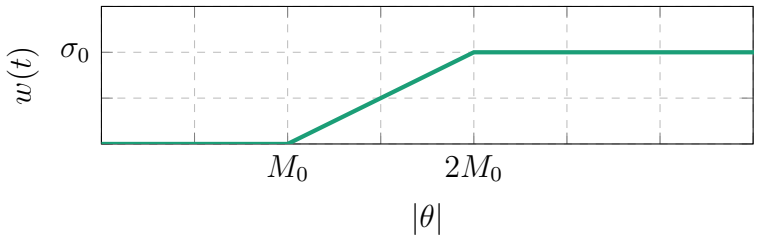


Figure 7.29: Continuous switching-σ function.

The parameters $M_{0_{\theta_3}}$ and $M_{0_{c_0}}$ are designed by considering the equations for the ideal control parameters θ_3 and c_0. The ideal control parameters depend on m_q, ω_λ, ζ_λ and ω_m. Therefore, by changing the value of the parameters, the bandwidth of the reference model ω_m is effectively changed as m_q, ω_λ, and ζ_λ are fixed. The parameters $2M_{0_{\theta_3}}$ and $2M_{0_{c_0}}$ are therefore obtained by choosing an upper limit for the bandwidth of the reference model. The upper limit is chosen as three-quarters of the bandwidth of the pitch controller, i.e. $\omega_{m_{2\max}} = \frac{3 \cdot 4.41}{4}$. Therefore, the parameters $M_{0_{\theta_3}}$ and $M_{0_{c_0}}$ are chosen at half this limit at $\omega_{m_{\max}} = \frac{3 \cdot 4.41}{8}$. This yields

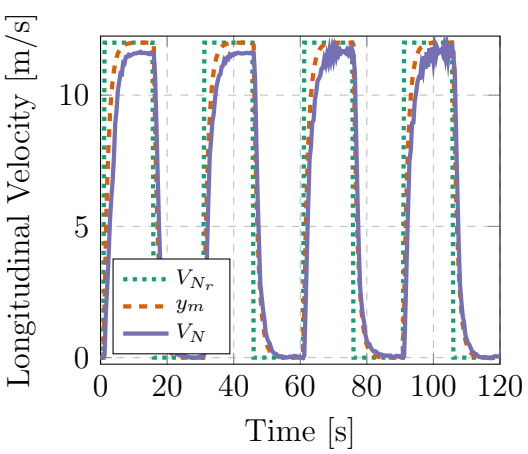
$$M_{0_{\theta_3}} = m_q(2\zeta_\lambda\omega_\lambda + \omega_{m_{\max}}), \text{ and} \tag{7.74}$$

$$M_{0_{c_0}} = m_q\omega_{m_{\max}}. \tag{7.75}$$

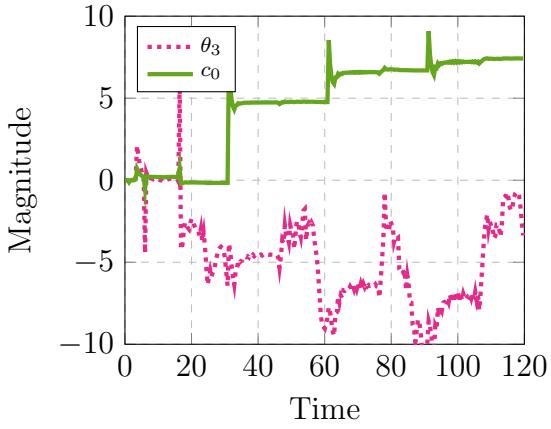
The term σ_0 was obtained through an iterative simulation process and the result is obtained as

$$\sigma_0 = 0.02. \tag{7.76}$$

The effects of sensor noise and external disturbances on the system is shown in Fig. 7.30 for a payload of 2 kg and a 1 m rod. High-frequency sensor noise and low-frequency sensor drift are implemented. Large velocity references were commanded to increase the effect of aerodynamic drag on the system, which serves as the external disturbance. The improvement that the leakage term provides is shown in Fig. 7.31.

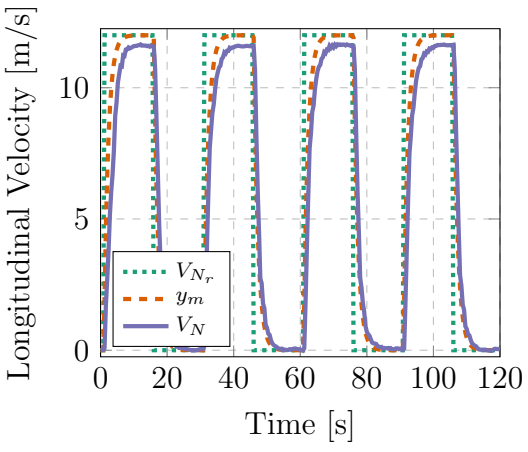


(a) Step response of the system.

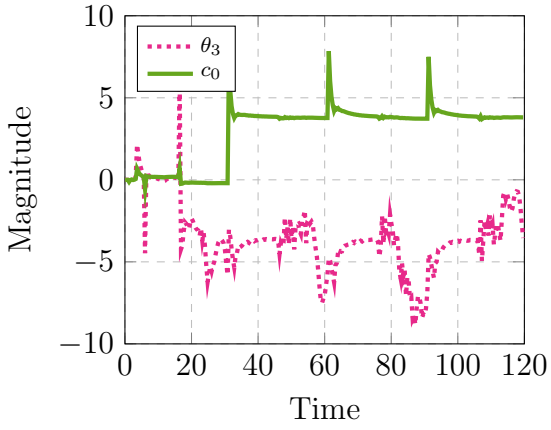


(b) Change in adaptive parameters from their initial values.

Figure 7.30: MRAC without leakage.



(a) Step response of the system.



(b) Change in adaptive parameters from their initial values.

Figure 7.31: MRAC with leakage.

Without the addition of leakage, the adaptive parameters continue to adapt to such an extent that additional oscillations are induced into the system. The effect of the leakage term is seen in the adaptive parameter plots, where c_0 is not allowed to continually increase and θ_3 does not drift as far from its initial value.

Leakage is sometimes referred to as a "*soft-bound*" as it does not put a limit on the parameters, but act in such a way to try and keep the parameters within some acceptable bounds. The spikes in the adaptive parameter plots can lead to unwanted results. Therefore, hard parameter bounds are also implemented to ensure that the parameters never exceed a certain upper and

lower limit. The adaptive law changes to

$$\frac{d\hat{\boldsymbol{\theta}}}{dt} = \begin{cases} -\boldsymbol{\Gamma}\text{sgn}\left(\frac{k_p}{k_m}\right)\left[\epsilon\boldsymbol{\phi} + w\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0\right)\right] & \text{if } |\hat{\boldsymbol{\theta}}| \leq |\boldsymbol{N}_0| \\ 0 & \text{if } |\hat{\boldsymbol{\theta}}| > |\boldsymbol{N}_0| \end{cases}. \tag{7.77}$$

The design of the bound $\boldsymbol{N}_0$ follows the same approach as with M_0. In this case, the limit is chosen as half of the bandwidth of the pitch controller. This yields

$$N_{0_{\theta_3}} = m_q(2\zeta_\lambda\omega_\lambda + \frac{4.41}{2}), \text{ and} \tag{7.78}$$

$$N_{0_{c_0}} = m_q\frac{4.41}{2}. \tag{7.79}$$

This concludes the improvements made to the adaptive law to keep the adaptive parameters from diverging when dealing with a practical system. The last problem to address is the presence of external disturbances. As shown in Fig. 7.31, a steady-state error exists when an external disturbance is present.

Sun et al. [48] explored a performance improvement of MRAC which improves both the transient and steady-state response. The modification adds a term to the control law, resulting in

$$u_p = \hat{\boldsymbol{\theta}}_1^T\frac{\boldsymbol{\alpha}(s)}{\Lambda(s)}u_p + \hat{\boldsymbol{\theta}}_2^T\frac{\boldsymbol{\alpha}(s)}{\Lambda(s)}y_p + \hat{\theta}_3 y_p + \hat{c}_0 r + u_a, \text{ where} \tag{7.80}$$

$$u_a = -C_a(s)e_1, \text{ and} \tag{7.81}$$

$$C_a(s) = \frac{\hat{c}_0 W_m^{-1}(s)}{(\tau_c s + 1)^{n^*} - 1} \tag{7.82}$$

$$= \frac{\hat{c}_0}{\omega_m}\cdot\frac{s + \omega_m}{\tau_c s}. \tag{7.83}$$

The term u_a is proportional to the error between the plant and reference model, which results in transient improvements when the error is large and zero steady-state tracking performance when the error is small. The time-constant τ_c is a parameter to be designed and should be chosen much faster than the reference model to not affect the longitudinal velocity dynamics. Therefore, it is chosen as

$$\tau_c = 0.4 \tag{7.84}$$

which is 3.125 times faster than the time-constant of the reference model. A step response is shown in Fig. 7.32 for a payload of 2 kg and a 1 m rod, with a constant disturbance introduced at time $t = 40$ s. The change in the adaptive parameters are shown in Fig. 7.33.

The controller quickly rejects the constant external disturbance and the steady-state tracking error returns to zero. The transient response is also improved with the modified MRAC as the response follows the reference model more accurately, yielding a smaller error e_1. The term u_a is responsible for this as it is proportional to the error, resulting in a reduction of e_1. The term u_a acquires an offset when the disturbance is introduced to reject its effect. The adaptive parameters θ_3 and c_0 do not adapt as much with the contribution of u_a. The parameters θ_3 and c_0 are made robust in the linear case and u_a absorbs some of the non-linearities of the system. This results in less adaptation which is an advantage of the modified MRAC.

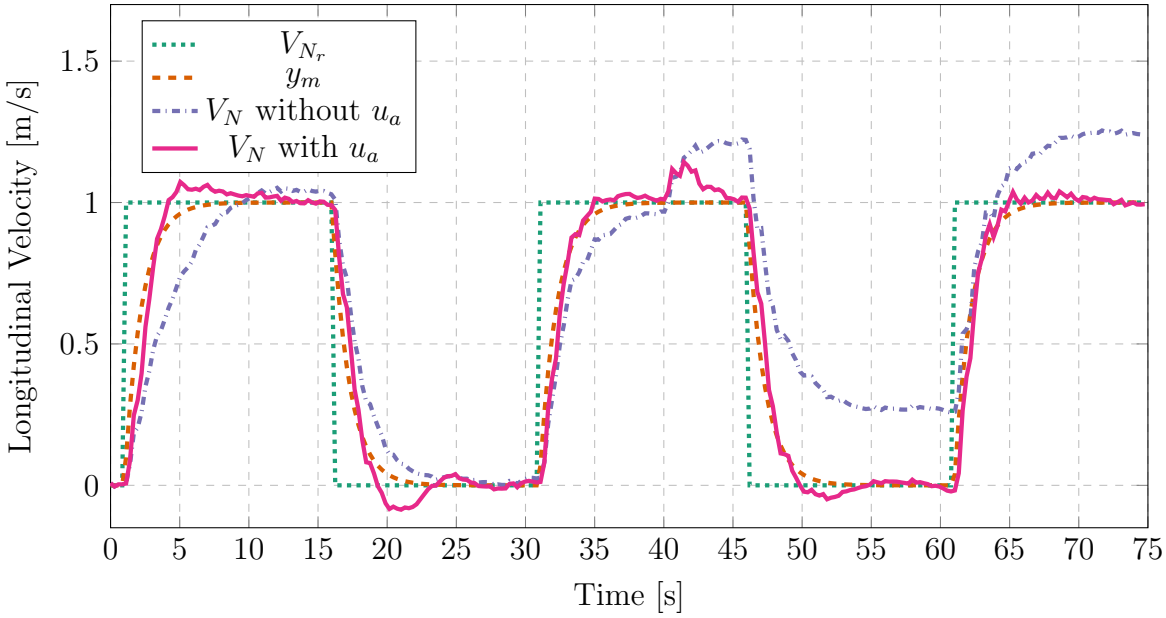


Figure 7.32: The modified MRAC longitudinal velocity response with sensor noise and a disturbance at $t = 40$ s.

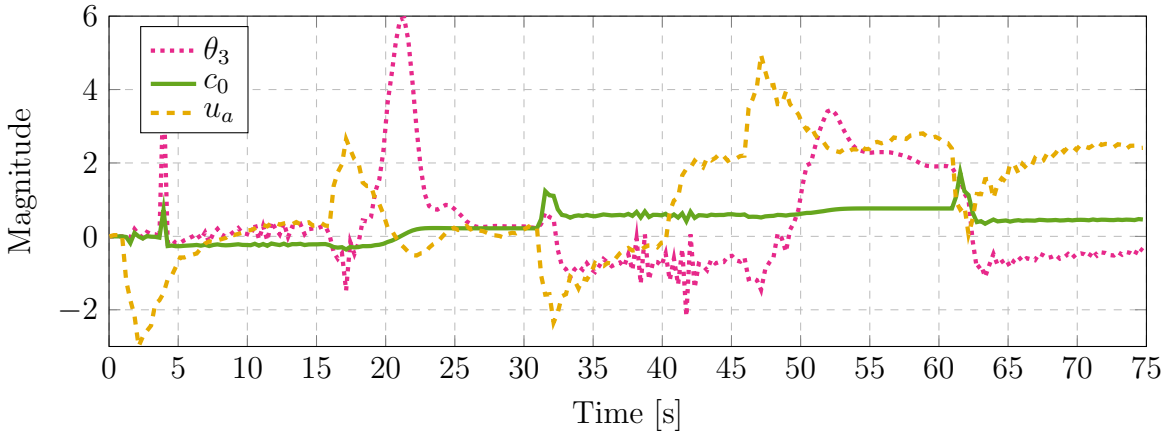


Figure 7.33: The change in the adaptive parameters and u_a from their initial values.

7.4.4 Summary

This approach makes use of the MRAC architecture to adapt the closed-loop response to follow a predefined reference model. The MRAC architecture consists of a control law and an adaptive law. The initial values of the adaptive parameters are made robust against payload parameter uncertainty. The adaptive law is responsible to adapt these parameters and techniques are implemented to keep the parameters within acceptable bounds. A modified MRAC control law was implemented to improve both the transient and steady-state response. This approach proves to damp the oscillations caused by the payload and can adapt to the specific payload attached to the vehicle.

It was found in simulation that this approach requires more control energy than the standard PID controller. This is mainly attributed to the spikes seen in the adaptive parameters. These spikes also appear in the pitch angle reference, resulting in a reference signal that is more noisy.

This reference signal, therefore, requires more control energy to perform the desired maneuvers.

For this approach, the position controller needs to be designed according to the reference model. The closed-loop linear longitudinal velocity dynamics will always tend towards the reference model for different payloads. As a result, the position controller can be a static controller and does not need to adapt for different payloads. This approach yields predictable results, even for different payloads.

7.5 Comparison and Summary

This chapter explored two approaches to solve the quadrotor with an unknown suspended payload problem. The first approach is based on an LQG controller which relies on online estimates of the payload mass and cable length. The second approach is based on the MRAC architecture. Both approaches proved to damp the oscillations caused by the payload and caters for the specific payload attached to the vehicle. These approaches are compared to one another and one is chosen to be implemented on a physical vehicle. The comparison is shown in Table 7.1.

	LQG Approach	**MRAC Approach**
1.	Consists of a lot of different algorithms, namely RLS, FFT, EKF, and LQR.	It is one large complex algorithm.
2.	The performance of each algorithm relies on the accuracy of the previous one.	The control and adaptive laws are designed separately and the performance mainly depends on the accuracy of the adaptive law.
3.	An initialization procedure with a state-machine is required.	The initial adaptive parameters are designed to be robust.
4.	Some components need to run at different rates, such as the EKF needs to run at a faster rate than the LQR controller.	The whole algorithm runs at the same rate.
5.	Obtains estimates of the payload parameters which can be validated online.	Techniques are implemented to keep the adaptive parameters from exceeding certain bounds.
6.	The control energy is also optimized by the LQR algorithm.	More control energy is required.
7.	The swing angle is directly minimized as an estimate is available.	The swing angle is indirectly minimized by the oscillations present in the velocity estimate.
8.	The position controller needs to be re-designed online.	A static position controller can be used.

Table 7.1: Comparison of the LQG approach and the MRAC approach.

It was decided to implement the adaptive controller for the practical vehicle. It has more advantages such as it is simpler to implement, does not require a state-machine or a re-design of the position controller, and is more predictable when considering different payloads. Therefore, it will be implemented in PX4, simulated with the PX4-Gazebo environment and demonstrated on a practical vehicle.

8. Practical Implementation and Results

This chapter focusses on the implementation and demonstration of the MRAC algorithm, developed in Chapter 7, in the PX4 environment. Firstly, the implementation details are given, followed by the SIL and HIL results of the PX4-Gazebo simulation. Thereafter, the practical flight test procedure and results are presented and discussed. Lastly, modifications are made to the flight control system based on observations from the practical flight test.

8.1 PX4 Implementation

The longitudinal linear velocity controller of PX4 is changed to incorporate the adaptive controller. A parameter is created to activate or deactivate the adaptive controller during flight. The activation of the adaptive controller switches from the PID longitudinal velocity controller to the MRAC scheme and changes the proportional gain of the position controller.

The control law and adaptive law are implemented in PX4 through the equations presented in Chapter 7. The MRAC algorithm filters some signals through the transfer functions $\frac{\boldsymbol{\alpha}(s)}{\Lambda(s)}$, $W_m(s)$, and $C_a(s)$. These transfer functions are descritezed with the bilinear transform, given as

$$s = \frac{2}{T_s} \cdot \frac{z-1}{z+1}, \tag{8.1}$$

where T_s is the sample time which corresponds to the frequency at which the velocity controllers are run, i.e. $T_s = 0.02$ s. The discrete transfer functions $\frac{\boldsymbol{\alpha}(z)}{\Lambda(z)}$, $W_m(z)$, and $C_a(z)$ are obtained and re-written as difference equations to be implemented.

The integration of the differential equation of the adaptive parameters, given by Eq. (7.64), is performed with standard Euler integration as T_s is quite small. The adaptive parameters are logged by PX4 to allow offline analysis of the performance of the adaptive controller.

The control gain of the position controller needs to be updated as the linear velocity controller has changed. The closed-loop dynamics of the longitudinal velocity controller is given by the MRAC reference model $W_m(s)$. Therefore, the plant of the north position dynamics is the integral of the reference model $W_m(s)$. The same design procedure is followed as in Section 6.1.8. The root locus of the combined plant and updated controller is shown in Fig. 8.1 and the step response is shown in Fig. 8.2.

The system response has no overshoot, a 2% settling time of 11.4 s and a bandwidth of 0.336 rad/s. The value of the proportional gain is 0.25. The updated gain is implemented in PX4 to be used with the adaptive longitudinal velocity controller.

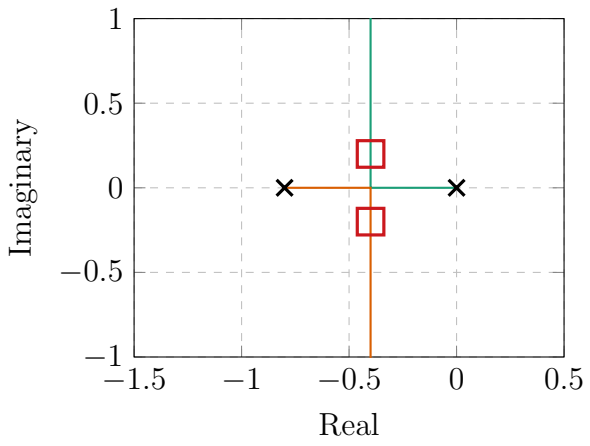


Figure 8.1: Root locus of the updated north position controller.

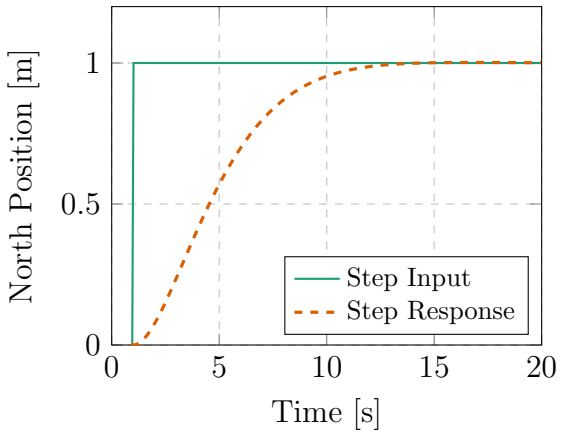


Figure 8.2: Step response of the updated north position controller.

8.2 PX4-Gazebo Simulation Results

The implemented adaptive controller is simulated with the PX4-Gazebo environment. A SIL simulation is performed to test the performance of the implemented adaptive controller and a HIL simulation is performed to test whether the algorithm can run on the designated hardware.

8.2.1 Software-in-the-Loop

Position waypoints are given to the vehicle to follow. After the first set of waypoints are performed with the standard quadrotor controllers designed in Chapter 6, the adaptive controller is activated at time $t = 43$ s. The result of the position response is shown in Fig. 8.3 and the velocity response is shown in Fig. 8.4 with a payload of 2 kg and a 1 m rod.

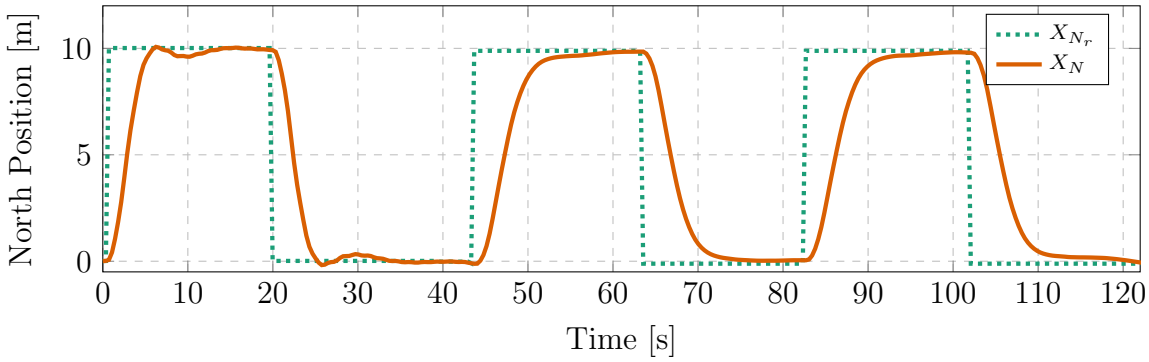


Figure 8.3: PX4-Gazebo simulation results of the north position response.

The adaptive controller damps the oscillations caused by the payload, which are clearly seen in the response before the adaptive controller is activated. The response sufficiently follows the reference model. The adaptation is difficult to see in the velocity response without square-wave inputs, but it is clear from the adaptive parameters, shown in Fig. 8.5 that adaptation takes place to accommodate the specific payload.

The spikes in the u_a term reduces the error between the output and the reference model during the transient state and the constant offset of the u_a term is to maintain a zero steady-state

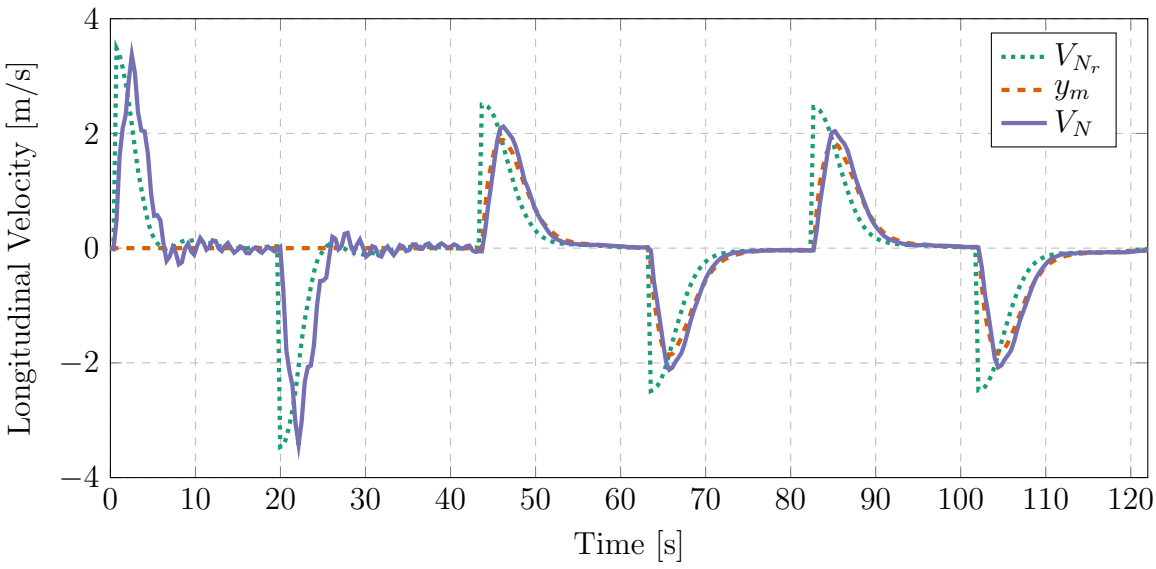


Figure 8.4: PX4-Gazebo simulation results of the longitudinal velocity response.

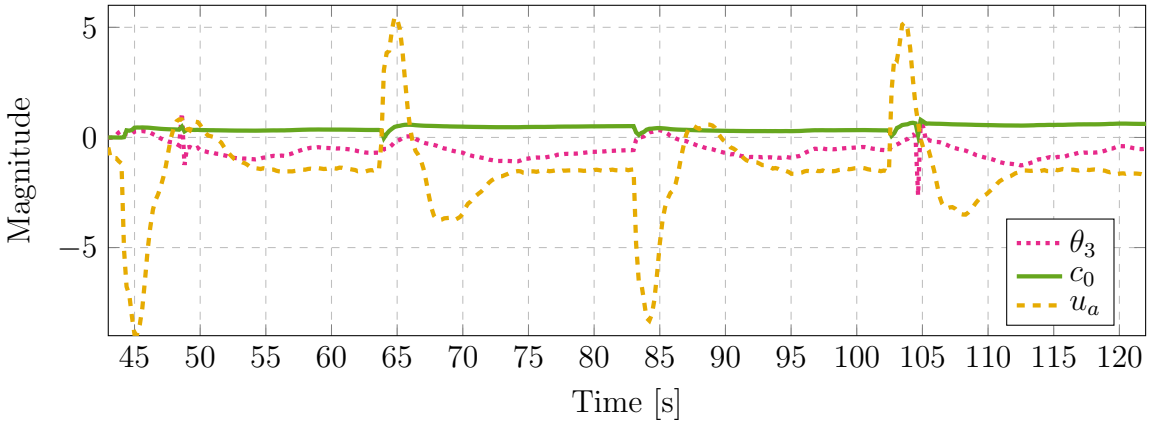


Figure 8.5: The change in the adaptive parameters from their initial values during the PX4-Gazebo simulation.

tracking error in the presence of disturbances and sensor biases. The parameters θ_3 and c_0 adapt in the same way as seen during the design of the MRAC algorithm.

The implemented MRAC algorithm works in PX4 as expected without any performance loss due to discretization.

8.2.2 Hardware-in-the-Loop

The HIL simulation tests the control system on the designated flight hardware using the same Gazebo simulator. The flight controller processor is a 32-bit STM32F427 Cortex M4. The Central Processing Unit (CPU) and Random Access Memory (RAM) specifications are:

- 168 MHz clock speed
- 256 kB Synchronous Dynamic RAM (SDRAM)

The HIL simulation results are the same as those shown in the SIL simulation and are not presented here. No mathematical differences are observed when performed by the flight con-

troller processor. However, the results of interest from the HIL simulation are the CPU load and RAM usage. These statistics indicate the processing power needed by the implemented MRAC algorithm. They are shown in Fig. 8.6 where the adaptive controller is activated at time $t = 43$ s.

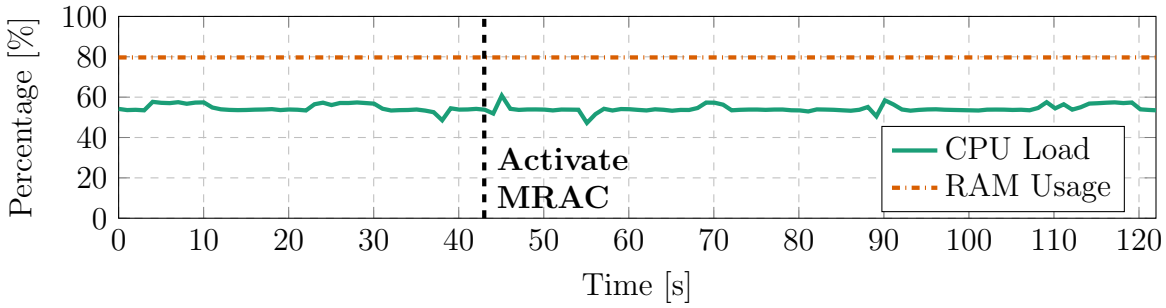


Figure 8.6: The CPU load and RAM usage of the Pixhawk during a HIL simulation.

The CPU load has an average value of about 55% and the RAM usage has an average value of about 80%. Both the CPU load and RAM usage does not indicate any increase when the adaptive controller is activated. Therefore, the algorithm has a negligible impact on the processing power of the Pixhawk avionics.

8.3 Practical Flight Tests

A series of practical flight tests were performed, shown in Fig. 8.7, to demonstrate the effectiveness of the proposed MRAC scheme in a practical flight. Three test flights were executed, each with a different payload. The different payloads are:

- No payload
- 1 kg payload with a 1 m rod
- 2 kg payload with a 1 m rod

Figure 8.7: Practical flight test of the quadrotor with a suspended payload.

For each of these payloads a tuned PID velocity controller is designed such as the one in Section 7.2 to serve as a baseline. The MRAC results will then be compared to the baseline result to analyze the performance of the MRAC scheme in the practical flight tests. For the case with no payload, the standard quadrotor longitudinal velocity controller, designed in Chapter 6, is used as the baseline. The tuned PID controller for the 2 kg payload is designed in Section 7.2 and the same procedure is followed for the 1 kg payload.

8.3.1 Flight Results

The practical flights consisted of performing multiple 10 m position steps. Firstly, position step inputs were commanded with the tuned PID controller active. Thereafter, the MRAC scheme is activated and the position step commands are repeated. The response of the tuned PID controller is shown in Fig. 8.8 and the response of the MRAC scheme is shown in Fig. 8.9 for the flight with the 1 kg payload. The measured payload swing angle of the flight, for the respective cases, is shown in Fig. 8.10.

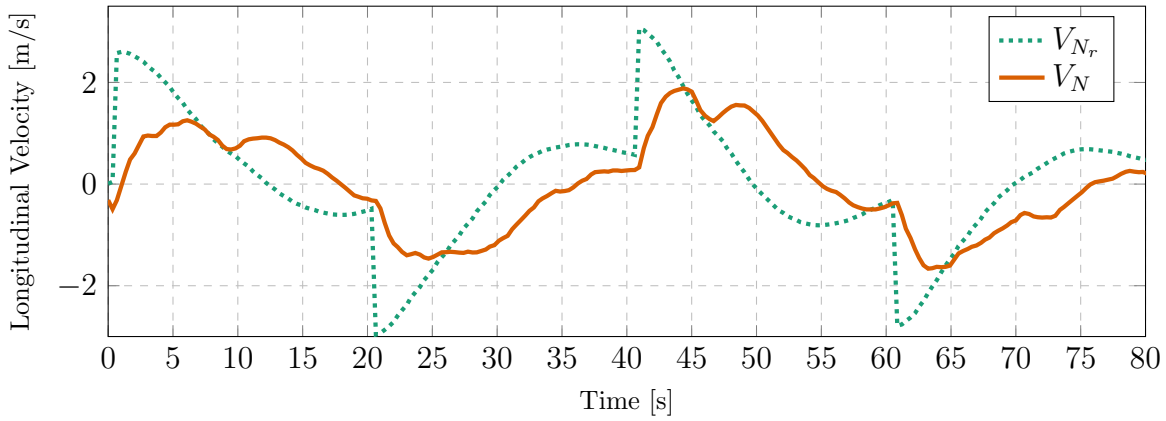


Figure 8.8: The practical response of the tuned PID controller with a 1 kg payload.

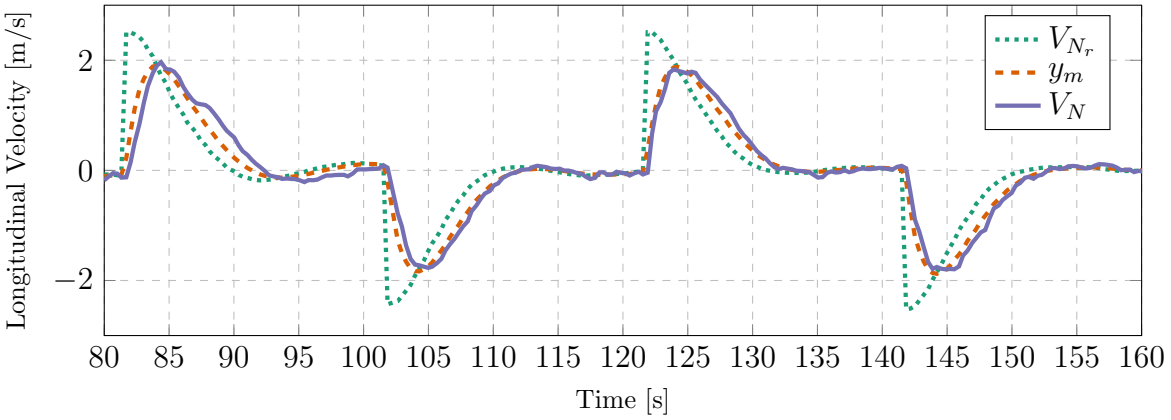
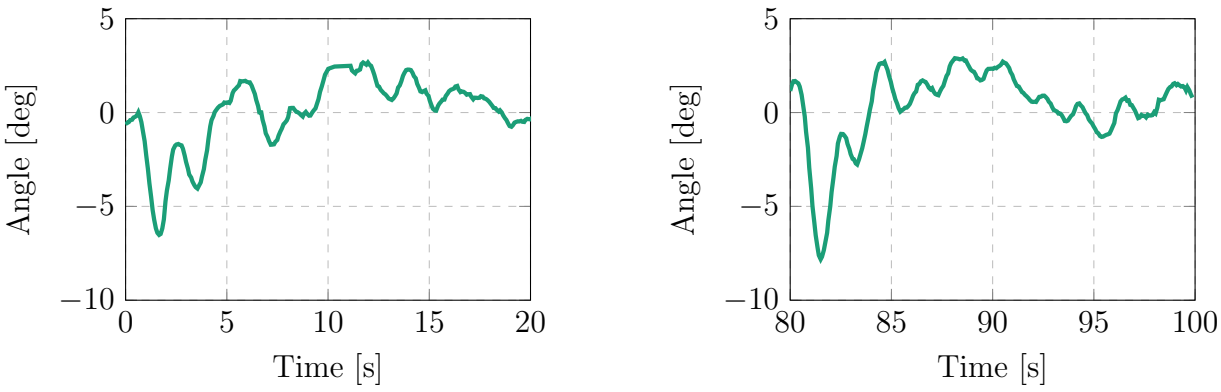


Figure 8.9: The practical response of the MRAC scheme with a 1 kg payload.

Both the tuned PID controller and the MRAC scheme are able to damp the oscillations caused by the payload. This is evident as the longitudinal velocity response contains no oscillations and the swing angle remains small. The MRAC response performs much better than the PID response in terms of reference following. This is because the u_a term of the MRAC



(a) PID response

(b) MRAC response

Figure 8.10: The measured payload angle of the practical flight with the 1 kg payload.

algorithm reduces the steady-state tracking error much faster than the integrator term of the PID controller. However, the MRAC scheme requires more control energy as it commands more aggressive pitch angle references to achieve the zero steady-state tracking error.

The error e_1 between the response and reference model is reduced from the first step response to the subsequent step responses. This indicates that adaptation takes place and is also clearly seen in the adaptive parameters, shown in Fig. 8.11. The c_0 term settles at a new value just after time $t = 100$ s, which is the same time at which the error e_1 is greatly reduced. The adaptive parameters of the practical flight adapt more than that of the simulation results. This is due to the presence of larger sensor noise and drift, causing the adaptive controller to react more aggressively to be able to follow the reference model.

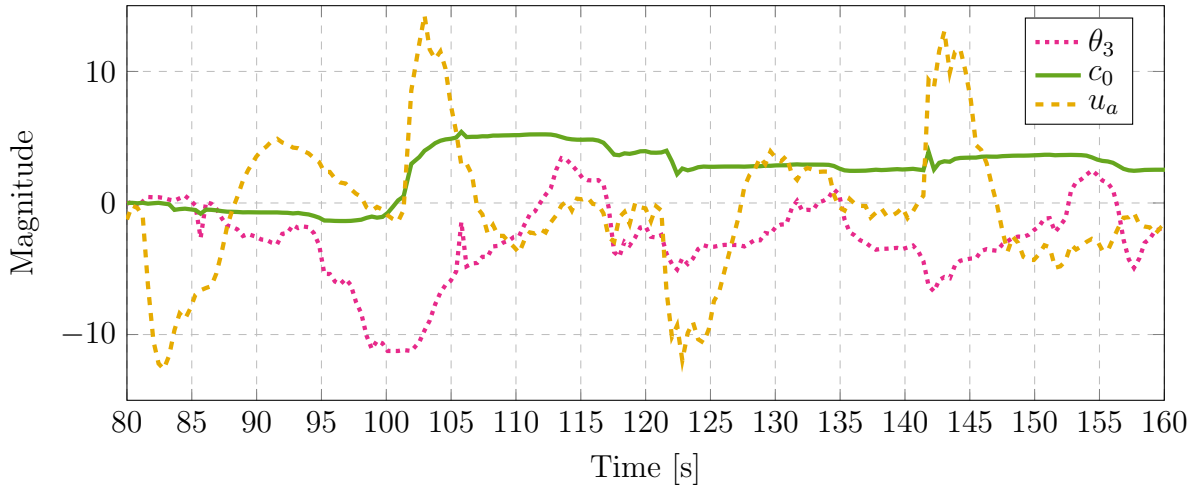


Figure 8.11: The change in the adaptive parameters from their initial values of the practical flight with the 1 kg payload.

A step response from each flight with the adaptive controller activated is shown in Fig. 8.12. The MRAC scheme provides consistent performance, regardless of the payload. The algorithm successfully changes the different closed-loop systems to that of the reference model. As mentioned, the consistent performance advantage allows the system to have one static position controller.

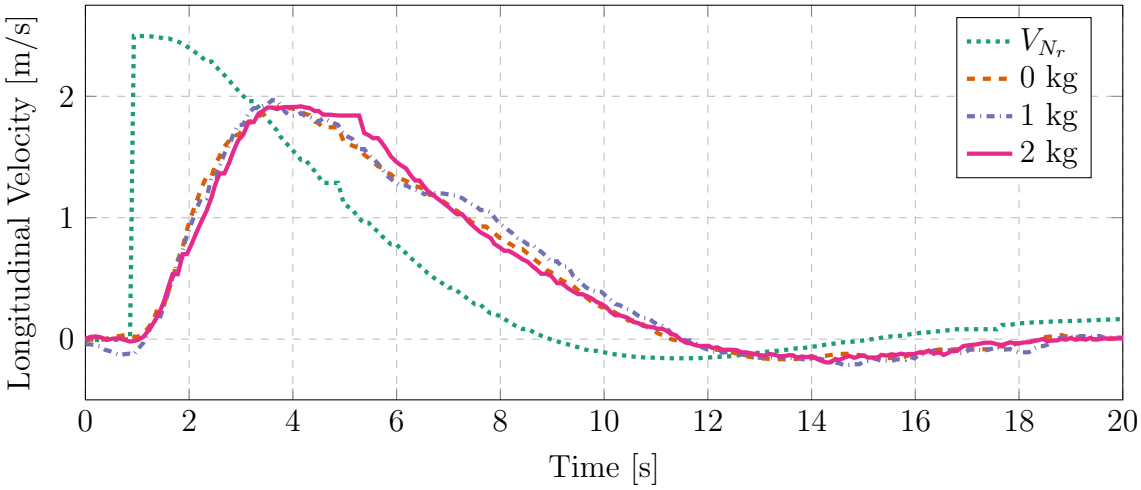


Figure 8.12: Comparison of the practical response for different payloads.

The practical flight tests were successful and the MRAC scheme performs as expected. The MRAC scheme for the unknown payload case can damp the payload oscillations just as well as the tuned PID controller for a known payload. The practical results are the confirmation that the main objective of this project was achieved.

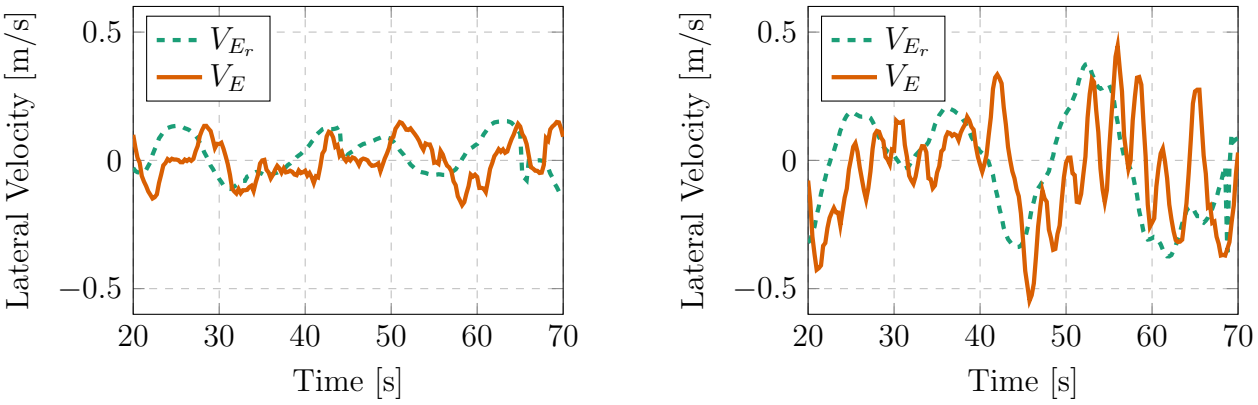
8.3.2 Out-of-Plane Oscillations

During the practical flight, it was observed that the rod flexed with the heavier payloads, causing out-of-plane oscillations. Therefore, the assumption that the rod is rigid did not hold during the practical flight tests. The flex of the rod is seen in Fig. 8.14 in comparison with Fig. 8.13 that contains no flex, which was taken from a bottom-mounted camera during the flight.

Figure 8.13: The rod without any flex during a practical flight test.

Figure 8.14: The flex of the rod during a practical flight test.

The flex caused unwanted out-of-plane oscillations. The lateral velocity response with no payload is shown in Fig. 8.15a and the response with the 2 kg payload is shown in Fig. 8.15b. The out-of-plane oscillations are clearly seen in the lateral velocity response of the flight with the 2 kg payload.

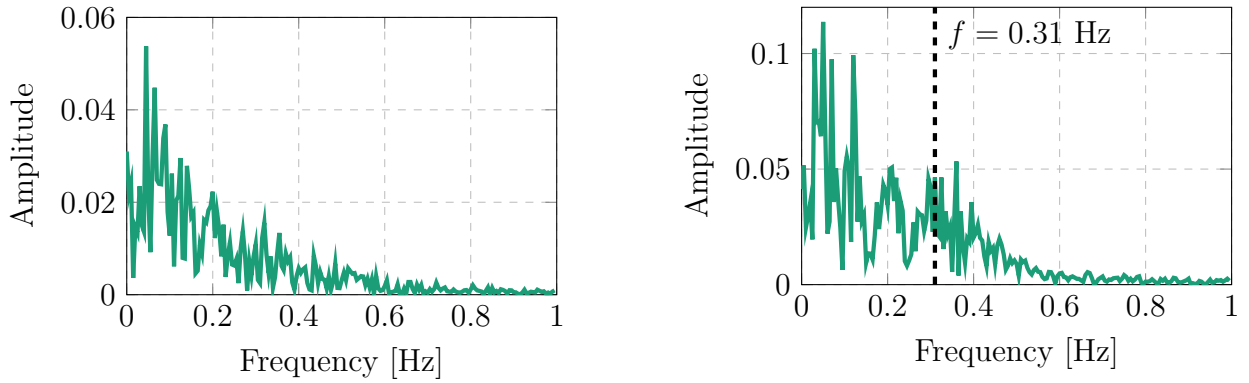


(a) No payload

(b) 2 kg payload

Figure 8.15: The practical lateral velocity response with different payloads.

The FFT of the lateral velocity response with no payload and with the 2 kg payload is shown in Fig. 8.16a and Fig. 8.16b, respectively. The frequency at $f = 0.31$ Hz is due to the oscillations caused by the payload in the lateral axis, which is not present in the FFT of the response with no payload.



(a) No payload

(b) 2 kg payload

Figure 8.16: The FFT of the practical lateral velocity response with different payloads.

A follow-up experiment was designed to attempt to damp these oscillations. The strategy is to use the principle of superposition to duplicate the MRAC algorithm in the longitudinal axis to the lateral axis. The MRAC scheme proved to damp the oscillations in the longitudinal axis and it is therefore applied in the lateral axis to attempt to damp the unwanted out-of-plane oscillations. In doing so, the follow-up experiment is a first attempt to solve the problem of a quadrotor carrying an unknown suspended payload able to swing in both axes.

The same MRAC algorithm is implemented in PX4 in the lateral axis. The Gazebo model is updated to allow the payload to swing in both directions, as shown in Fig. 8.17, to test the algorithm in simulation. The simulation results of the inertial positions and linear velocities are shown in Fig. 8.18 and Fig. 8.19, respectively, of a quadrotor carrying a 2 kg payload with a 1 m rod. Waypoints were commanded in both the north and east positions to allow the payload to swing in both directions independently. The MRAC algorithm was activated in both directions at time $t = 45$ s.

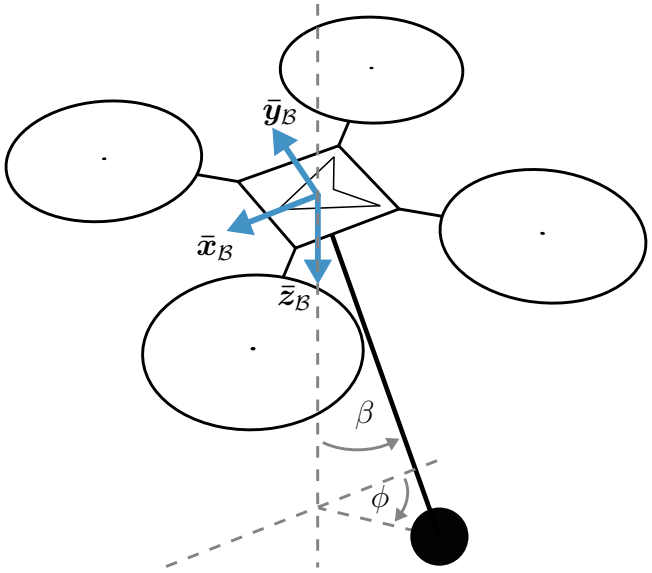


Figure 8.17: A quadrotor carrying a suspended payload swinging in both directions.

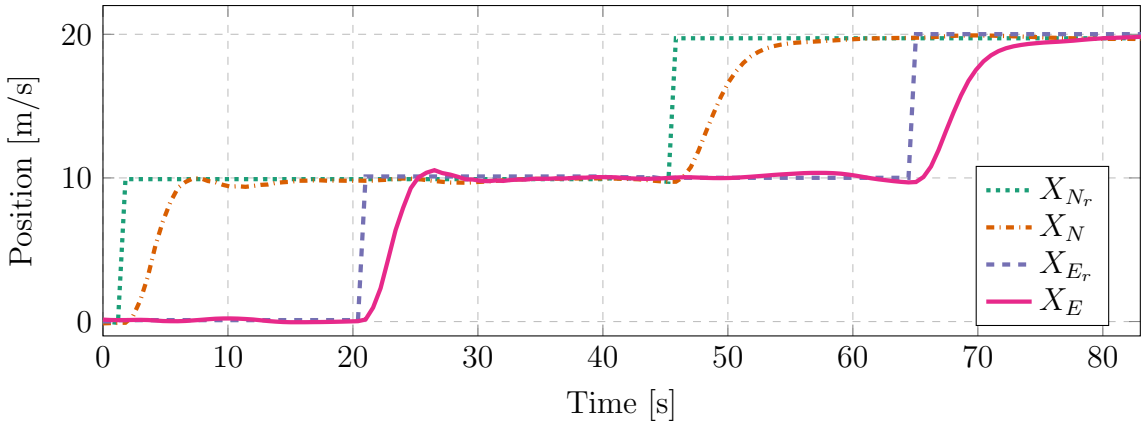


Figure 8.18: The PX4-Gazebo position simulation result with the lateral MRAC algorithm.

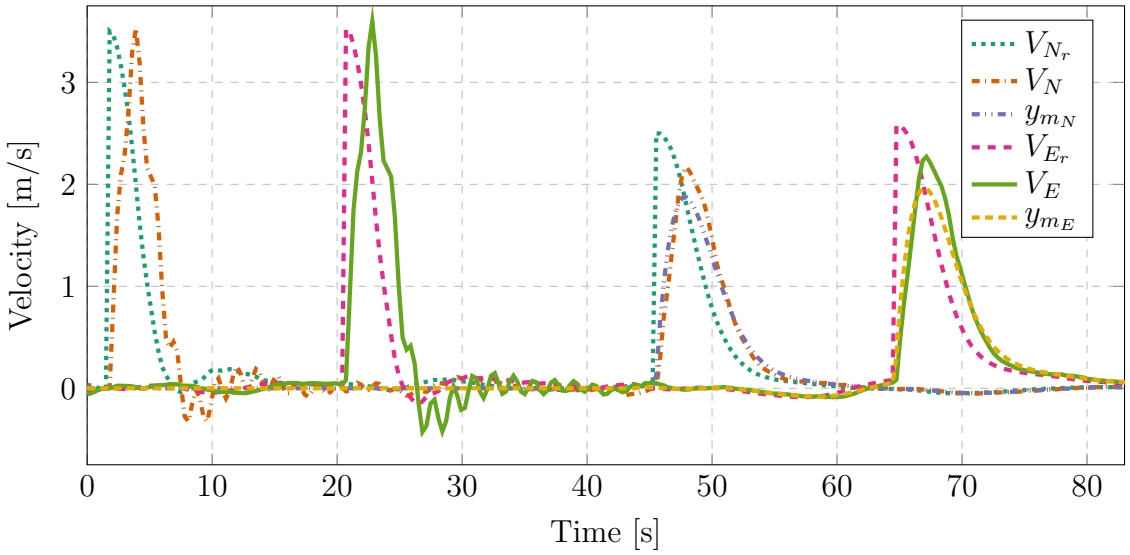
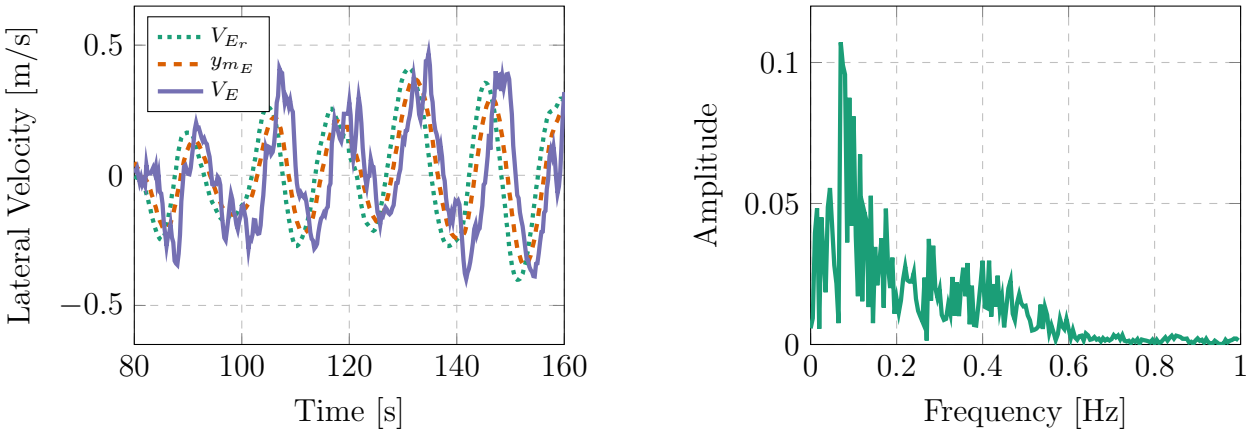


Figure 8.19: The PX4-Gazebo velocity simulation result with the lateral MRAC algorithm.

The simulation results prove that the MRAC algorithm is able to damp the oscillations present in both the longitudinal and lateral directions. The principle of superposition is successfully applied to the problem and the solution is able to damp the oscillations caused by an unknown suspended payload able to swing in both axes.

The practical flight test with the 2 kg payload was repeated with the addition of the implemented MRAC algorithm in the lateral axis. The goal was to reduce the unwanted out-of-plane oscillations while damping the longitudinal oscillations induced when performing north position step responses. The lateral velocity response of the practical flight is shown in Fig. 8.20a and the FFT of the response is shown in Fig. 8.20b.



(a) The velocity response.

(b) The FFT of the response.

Figure 8.20: The practical response with the lateral MRAC scheme.

The 0.31 Hz oscillations caused by the payload are damped. These oscillations are not seen in the lateral velocity response and are not present in the FFT as it was previously in Fig. 8.16b. However, the commanded velocity setpoints are large and slowly oscillating. This is due to the suspended payload joint, see Fig. 4.3, which is not able to rotate in the lateral axis. Therefore, when the quadrotor performs a roll maneuver to counter an oscillation, the heavy payload causes the rod to flex. This is exaggeradly illustrated in Fig. 8.21. The flex applies a force on the vehicle, which is seen as a disturbance to be rejected, causing the vehicle to eventually roll in the other direction. This causes the large slowly oscillating response in the lateral velocity.

Figure 8.21: Illustration of payload rod flex during a roll maneuver.

The implemented MRAC algorithm in the lateral axis was able to damp the out-of-plane oscillations caused by the payload. However, the roll maneuvers cause more flex in the payload rod which leads to unwanted oscillatory dynamics. Therefore, it is concluded that the lateral MRAC scheme performs as expected, but the setup of the practical flight test need to be adapted. The joint of the suspended payload should be replaced with one allowing the payload to swing in both axes. This may lead to more predictable results than the more complex bending dynamics of the rod. A quadrotor with a payload able to swing in both axes was simulated in the PX4-Gazebo environment and produced good results. The practical setup needs to change accordingly to produce results of the same standard.

8.4 Summary

This chapter focused on the implementation and practical demonstration of the designed MRAC algorithm. The algorithm was implemented in PX4 and simulated in the PX4-Gazebo environment. The simulation produced good results where the oscillations caused by the payload were sufficiently damped.

Practical test flights were performed to test the effectiveness of the proposed algorithm on a physical quadrotor vehicle. The performance of the MRAC scheme was consistent during practical flights with different payloads. The MRAC response performed better than a PID controller tuned for the specific payload.

During the practical flight tests, out-of-plane oscillations were observed with the heavier payloads. A follow-up flight test was performed with the MRAC algorithm duplicated in the lateral axis to reduce the out-of-plane oscillations. The oscillations were successfully reduced, but the payload joint induced unwanted dynamics on the vehicle during roll maneuvers. The joint of the suspended payload should change to allow the payload to swing in both axes. By doing this, the proposed solution will be able to damp the oscillations caused by a payload swinging in both axes.

9. Conclusion

This thesis addressed the problem of a quadrotor transporting an unknown suspended payload able to swing in one axis. The parameters of the suspended payload are unknown and its state is not available for measurement. This function enables any vehicle to transport a suspended payload. The oscillatory behavior of the payload significantly affects the flight dynamics of the vehicle. A solution is required to damp these oscillations and ensure stable flight of the vehicle.

9.1 Current Solutions in Literature

A literature study was done to identify the current solutions to the problem. The current solutions either address the problem with unknown payload parameters or the problem with unknown payload states. This project attempts to contribute to the area with both unknown payload parameters and states. For an unknown suspended payload, two trends were identified regarding the current solutions:

- Estimators are used to estimate either the payload parameters or payload states. These estimates are used in the control system.
- Adaptive control techniques are used to simultaneously estimate and control the unknown payload.

These trends inspired the proposed solution for this project.

9.2 Proposed Solution

The proposed solution attempts to damp the oscillations caused by the unknown suspended payload while maintaining stable flight. It was assumed that the suspended payload is attached to the CoM of the vehicle. Therefore, the payload only affects the translational dynamics and not the rotational dynamics of the vehicle. The linear velocity controller of the vehicle should, therefore, be responsible to damp the oscillations. Two different methods were explored to attempt to achieve this. These methods are:

- An LQG control approach.
- An adaptive control approach.

Both of these methods were explored in simulation. Their results and details were compared in order to choose one technique to implement on a practical vehicle.

A custom quadrotor vehicle was built for the practical demonstration of the proposed solution. A Pixhawk flight controller was used, running the PX4 flight control stack firmware. The PX4 codebase was thoroughly studied and the control architecture was identified. It was decided to use the architecture and change the control gains according to the custom-built quadrotor.

A model of the custom-built quadrotor was derived and implemented in MATLAB/Simulink to design the control gains. Thereafter, a model of the vehicle was implemented in the Gazebo simulator to perform SIL and HIL simulations with the PX4-Gazebo environment. Flight tests were performed to ensure that the vehicle maintains stable flight with the designed controllers. The flight tests were successful and the results closely resemble those obtained in simulation. It was then concluded that the simulation environment is a good representation of the physical system.

The two control strategies for the quadrotor with the unknown suspended payload were then explored, followed by the practical implementation and demonstration of the preferred solution.

9.2.1 LQG Controller

The first control method makes use of a full-state feedback controller to simultaneously control the velocity of the vehicle and damp the payload oscillations. Therefore, estimates of the payload parameters and state are needed. A sequence of operations is performed to independently estimate the needed parameters and state. A RLS algorithm is used in the vertical axis to estimate the payload mass. A position step input is then commanded to allow the payload to swing. The swing oscillations are present in the linear velocity measurement and a FFT is performed to obtain the oscillatory frequency. The length of the payload rod is proportional to the frequency of oscillation and is then calculated. Thereafter, an LQG controller is activated which consists of an EKF and an LQR controller. The EKF estimates the swing angle of the payload and the LQR controller makes use of full-state feedback to simultaneously control the velocity of the vehicle and the payload swing angle. This approach was explored in simulation. It provided sufficient damping and proved to be robust against sensor noise and external disturbances.

9.2.2 Adaptive Controller

The other control method makes use of the adaptive control scheme known as Model Reference Adaptive Control (MRAC). MRAC attempts to change the closed-loop dynamics of the system to that of a predefined reference model by adapting its control parameters. The algorithm consists of a control law and an adaptive law. The control law was explored and made robust against parameter uncertainty of the payload. The result is a stable response even before adaptation takes place. The response is stable but lacks the required performance and therefore an adaptive law was used to improve the control parameters to achieve the desired performance. The adaptive law adapts the control parameters based on the error between the response and the reference model. Phenomena such as sensor noise and external disturbances can lead to the divergence of the control parameters. Therefore, methods were explored and implemented to avoid this, which yielded good results. The simulation results of the MRAC algorithm proved to sufficiently damp the oscillations caused by the payload.

9.3 Practical Flight Tests

The adaptive control strategy was chosen to implement on a practical vehicle for flight tests. It is the preferred method as it is simpler to implement and it provides consistent performance for different payloads.

Different suspended payloads were attached to the quadrotor for the practical flight tests. Three flights were performed, each carrying a different payload, with:

- A 2 kg mass and a 1 m rod.
- A 1 kg mass and a 1 m rod.
- No payload.

Each flight consisted of several position step commands with a tuned PID controller and the MRAC algorithm. The PID controller is tuned for the specific attached payload to reduce the oscillations. It serves as a baseline to compare to the MRAC response. The MRAC algorithm proved to sufficiently damp the oscillations caused by the payload. The tuned PID controller also damped the payload oscillations, but the MRAC algorithm outperformed it in terms of reference following. The flight tests were successful in practically demonstrating the effectiveness of the MRAC scheme. It proved to perform consistently with the different payloads.

It was observed that the payload rod flexed with the heavier payloads, which caused out-of-plane oscillations. This project focused on the problem of a suspended payload able to swing in one axis. However, with the out-of-plane oscillations, the payload was able to induce oscillations in both axes. With this observation, a follow-up experiment was designed as a first attempt to solve the problem of a payload able to swing in both axes. Therefore, the principle of superposition was applied and the MRAC algorithm was duplicated in the lateral axis to attempt to damp the out-of-plane oscillations. It proved to damp the oscillations, but the joint of the payload can not rotate in the lateral axis, resulting in an increase of the flex of the rod during a roll maneuver. This resulted in unwanted flight dynamics. Therefore, the joint of the payload needs to change to allow the payload to swing in both directions, resulting in more predictable results. In doing so, practical flight results can be obtained with the current solution for a quadrotor transporting a suspended payload able to swing in both axes.

9.4 Future Work

Future work includes the continuation of this project to change the suspended payload to swing in both axes. The MRAC algorithm was applied in the lateral axis to allow the damping of oscillations in both axes. The performance of the algorithms can be explored with the coupled oscillatory behavior when commanding simultaneous longitudinal and lateral inputs. Possible ways to extend the practical setup and maintain the online logging of the ground truth swing angles are:

- Make use of a two-axis encoder to log measurements of the swing angles for offline analysis.
- Design a joint consisting of two potentiometers to log the swing angles.
- Make use of a downward-facing camera to estimate the position of the payload.

Future work also includes the practical demonstration of the LQG controller. This requires the implementation of the RLS, FFT, EKF and LQR algorithms. Each of these components can also be used separately in combination with other techniques to control the quadrotor and suspended payload. Possible future strategies include:

- The RLS and FFT algorithms for the payload mass and rod length estimation can be

combined with the adaptive controller to provide better initial estimates of the control parameters.

- The payload mass and rod length estimates can also be used to choose an appropriate reference model for the adaptive controller.
- The swing angle estimate from the EKF can also be used to implement a full-state feedback adaptive controller.
- The LQG controller can be used with the measurement of the payload swing angle for better performance as the ground-truth value is used instead of an estimate.
- The estimates obtained from the RLS, FFT and EKF algorithms can be used to implement a trajectory generation algorithm to minimize the swing angles.
- Model Predictive Control can be used to minimize the swing angles along with an adaptive component to adapt to the specific attached payload.
- The estimates obtained from the RLS, FFT and EKF algorithms can be used in a feedback linearization algorithm to reject the oscillatory effect of the payload on the vehicle.

It is clear that the methods used in this thesis can serve as a basis to propose other solutions to the problem.

References

[1] J. Plaza, "Exploring the collaboration between nasa, the faa and drone industry to develop a cloud-based utm system," 2019, Accessed: 2019-05-20. [Online]. Available: https://www.expouav.com/news/latest/exploring-the-collaboration-between-nasa-the-faa-and-drone-industry-to-develop-a-cloud-based-utm-system/

[2] Dan, "The top 10 most popular drones used by uav service providers on an industrial platform," 2018, Accessed: 2019-05-20. [Online]. Available: https://djm-aerial.com/top-ten-drones-for-industry/

[3] T. Godart, "Disruption at the edge: Iot transforming energy grids," 2018, Accessed: 2019-05-20. [Online]. Available: https://blogs.intel.com/iot/2018/01/16/disruption-at-the-edge-iot-transforming-energy-grids/#gs.bqfrp9

[4] D. Bell, "3d robotics' new solo quadcopter: Dual linux processors, unprecedented gopro control," 2015, Accessed: 2019-05-20. [Online]. Available: https://makezine.com/2015/04/13/3dr-solo-quadcopter-gopro/

[5] "Irrigation management drones." [Online]. Available: https://specdrones.us/agriculture/irrigation-management-drones/

[6] B. Mack, "Flirtey is making domino's pizza delivery by drone available for regular folks," 2017, Accessed: 2019-05-20. [Online]. Available: https://idealog.co.nz/tech/2017/06/flirtey-making-dominos-pizza-delivery-drone-available-regular-folks

[7] A. Hern, "Amazon claims first successful prime air drone delivery," 2016, Accessed: 2019-05-20. [Online]. Available: https://www.theguardian.com/technology/2016/dec/14/amazon-claims-first-successful-prime-air-drone-delivery

[8] ——, "Customers in australian capital can get food and coffee delivered via drone," 2019, Accessed: 2019-05-20. [Online]. Available: https://techcrunch.com/2019/04/09/customers-in-australian-capital-can-get-food-and-coffee-delivered-via-drone/

[9] M. McGee, "Amazon delivered its first order via drone," 2016, Accessed: 2019-05-20. [Online]. Available: https://marketingland.com/amazon-delivered-first-order-via-drone-200940

[10] "Googles wing is launching a public drone for delivery service in australia." [Online]. Available: https://www.ilounge.com/news/technology/googles-wing-is-launching-a-public-drone-for-delivery-service-in-australia

[11] K. Zraick, "Like uber for organs: drone delivers kidney to maryland woman," 2019, Accessed: 2019-05-20. [Online]. Available: https://www.nytimes.com/2019/04/30/health/drone-delivers-kidney.html

[12] L. Li, L. Sun, and J. Jin, "Survey of advances in control algorithms of quadrotor unmanned aerial vehicle," *International Conference on Communication Technology Proceedings, ICCT*, vol. 2016-February, pp. 107–111, 2016.

[13] S. Sadr, S. A. A. Moosavian, and P. Zarafshan, "Dynamics modeling and control of a quadrotor with swing load," *Journal of Robotics*, 2014.

[14] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation : Safe and efficient load manipulation with aerial robots," *IEEE Robotics and Automation Magazine*, vol. 19, pp. 69–79, 2012.

[15] "Amazon prime air," Accessed: 2019-05-20. [Online]. Available: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011

[16] J. Barret, "I, for one, welcome our new automated robot overlords," 2016, Accessed: 2019-05-20. [Online]. Available: https://www.digitaltrends.com/opinion/i-welcome-our-future-robot-overlords/

[17] B. Min, J. Hong, and E. Matson, "Adaptive robust control (ARC) for an altitude control of a quadrotor type UAV carrying an unknown payloads," *International Conference on Control, Automation and Systems*, pp. 1147–1151, 2011.

[18] B. J. Emran, J. Dias, L. Seneviratne, and G. Cai, "Robust adaptive control design for quadcopter payload add and drop applications," *Chinese Control Conference, CCC*, vol. 2015-September, pp. 3252–3257, 2015.

[19] X. Ma and H. Bao, "An anti-swing closed-loop control strategy for overhead cranes," *Applied Sciences*, vol. 8, p. 1463, 2018.

[20] Y. Qian, Y. Fang, and B. Lu, "Adaptive repetitive learning control for an offshore boom crane," *Automatica*, vol. 82, pp. 21–28, 2017.

[21] M. Bisgaard, A. la Cour-Harbo, and J. Dimon Bendtsen, "Adaptive control system for autonomous helicopter slung load operations," *Control Engineering Practice*, vol. 18, pp. 800–811, 2010.

[22] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2691–2697, 2012.

[23] S. Wang and B. Xian, "An anti-swing trajectory approach for an unmanned aerial vehicle with a slung payload," *Chinese Control Conference, CCC*, vol. 2018-July, pp. 5560–5565, 2018.

[24] Y. Alothman, W. Jasim, and D. Gu, "Quad-rotor lifting-transporting cable-suspended payloads control," *2015 21st International Conference on Automation and Computing: Automation, Computing and Manufacturing for New Economic Growth, ICAC 2015*, 2015.

[25] F. A. Goodarzi, D. Lee, and T. Lee, "Geometric control of a quadrotor UAV transporting a payload connected via flexible cable," *International Journal of Control, Automation and Systems*, vol. 13, pp. 1486–1498, 2015.

[26] S. Yang and B. Xian, "Robust control design for the quadrotor UAV with a suspended payload," *Proceedings of 2018 IEEE 8th Annual International Conference*, pp. 469–473, 2018.

[27] G. V. Raffo and M. M. De Almeida, "Nonlinear robust control of a quadrotor UAV for load transportation with swing improvement," *Proceedings of the American Control Conference*, vol. 2016-July, pp. 3156–3162, 2016.

[28] S. Dai, T. Lee, and D. S. Bernstein, "Adaptive control of a quadrotor UAV transporting a cable-suspended load with unknown mass," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2015-February, pp. 6149–6154, 2014.

[29] S. Yang and B. Xian, "Energy-based nonlinear adaptive control design for the quadrotor UAV system with a suspended payload," *IEEE Transactions on Industrial Electronics*, 2019.

[30] E. L. De Angelis, "Swing angle estimation for multicopter slung load applications," *Aerospace Science and Technology*, vol. 89, pp. 264–274, 2019.

[31] M. E. Guerrero-Sánchez, D. A. Mercado-Ravell, R. Lozano, and C. D. García-Beltrán, "Swing-attenuation for a quadrotor transporting a cable-suspended payload," *ISA Transactions*, vol. 68, pp. 433–449, 2017.

[32] H. Schaub, "Kinematics: Describing the motions of spacecraft," 2018, Accessed: 2018-06-29. [Online]. Available: https://www.coursera.org/learn/spacecraft-dynamics-kinematics

[33] J. Blakelock, *Automatic Control of Aircraft and Missiles*. Wiley, 1991. [Online]. Available: https://books.google.co.in/books?id=ubcczZUDCsMC

[34] P. D. S. Moller, "Automated Landing of a Quadrotor Unmanned Aerial Vehicle on a Translating Platform," 2015. [Online]. Available: https://scholar.sun.ac.za/handle/10019.1/98014

[35] A. A. Shabana, *Dynamics of multibody systems*. Cambridge University Press, 2013. [Online]. Available: https://books.google.co.za/books?id=CYlGAAAAQBAJ&source=gbs_book_similarbooks

[36] "Px4." [Online]. Available: https://github.com/PX4/Devguide

[37] "Ardupilot." [Online]. Available: https://github.com/ArduPilot/ardupilot_wiki

[38] "Betaflight." [Online]. Available: https://github.com/betaflight/betaflight

[39] "inav." [Online]. Available: https://github.com/iNavFlight/inav-configurator

[40] P. G. Ioppo, "The Design, Modelling and Control of an Autonomous Tethered Multirotor UAV," 2017. [Online]. Available: https://scholar.sun.ac.za/handle/10019.1/101095

[41] "Px4 developer guide." [Online]. Available: https://px4.io/developer-guide/

[42] Christoph, "PID controller basics eliminating derivative kick." [Online]. Available: https://barela.wordpress.com/2013/07/19/pid-controller-basics-eliminating-derivative-kick/

[43] A. Adiprasetya and A. S. Wibowo, "Implementation of PID controller and pre-filter to control non-linear ball and plate system," *ICCEREC 2016 - International Conference on Control, Electronics, Renewable Energy, and Communications 2016, Conference Proceedings*, 2017.

[44] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control," 2013.

[45] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011.

[46] K. Åström and B. Wittenmark, *Adaptive Control*. Dover Publications, 2008.

[47] P. Ioannou and J. Sun, *Robust Adaptive Control*. Dover Publications, 2012. [Online]. Available: http://www.amazon.com/Robust-Adaptive-Control-Electrical-Engineering/dp/0486498174

[48] J. Sun, A. Olbrot, and M. Polis, "Robust stabilization and robust performance using model reference control and modeling error compensation," *IEEE Transactions on Automatic Control*, 1994.

[49] T-Motor, "T-motor mn5212 electric motor." [Online]. Available: http://store-en.tmotor.com/goods.php?id=377

[50] W. Johnson, "Helicopter Theory," 1994.

A. Quadrotor Physical Parameter Calculations

A.1 Propulsion System Performance

The chosen propulsion system for the practical quadrotor consists of T-Motor MN5212 KV340 motors, T-Motor P18x6.1 propellers and T-Motor Air 40A ESCs. The performance of this propulsion system is obtained from [49], where the most important factors are shown in Table A.1.

Motor	Voltage (V)	Propeller	Throttle	Current (A)	Torque (Nm)	Thrust (kg)
MN5212 KV340	24	P18x6.1	50%	5.7	0.290	1.318
			55%	7.4	0.344	1.612
			60%	9.3	0.411	1.901
			65%	11.6	0.472	2.259
			75%	16.5	0.605	2.835
			85%	22.1	0.737	3.477
			100%	31.0	0.918	4.355

Table A.1: T-Motor MN5212 340KV performance.

A.2 Calculations of the Quadrotor Payload Capabilities

A desired thrust to weight ratio is $2:1$, which allows the vehicle to hover at 50% throttle and still have enough thrust to perform manuevers. With this in mind, the payload that the vehicle can carry is calculated as

$$m_{p_{max}} = \frac{4.355 \cdot 4}{2} - 4.555 = 4.155 \text{ kg}. \tag{A.1}$$

The total flight time of the vehicle and payload is calculated by using the capacity of the battery and the current draw information of the motor. The design includes two 5000 mAh LiPo batteries. Each battery can supply a constant 5000 mA for an hour. These batteries are run in parallel, doubling the capacity. With a payload of 4.155 kg, each motor needs to supply

$$T_{p_{hover}} = \frac{4.555 + 4.155}{4} = 2.178 \text{ kg} \tag{A.2}$$

of thrust. From Table A.1, the current draw of each motor at this thrust is about 11.6 A. The current consumption of the avionics is estimated as 1 A and the total flight time is then be calculated as

$$t_{p_{hover}} = \frac{5+5}{4 \cdot 11.6 + 1} \cdot 60 = 12.66 \text{ min}. \tag{A.3}$$

The same calculations are made without any payload, yielding a total flight time of about

$$t_{p_{hover}} = \frac{5+5}{4 \cdot 5.7 + 1} \cdot 60 = 25.21 \text{ min}. \tag{A.4}$$

A.3 Mass Moment of Inertia Experiment

The mass moment of inertia experiment, used in Section 4.2.1, is explained in this section. Consider the illustration of a disc with an unknown inertia hanging by two ropes, as shown in Fig. A.1. If this disc is perturbed, as shown in Fig. A.2, the disc will oscillate about the vertical

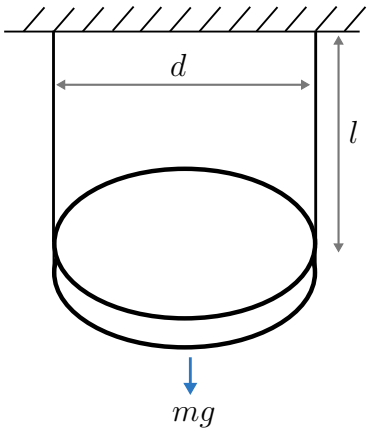


Figure A.1: Inertia experiment setup.

axis and eventually come to a stop due to friction. This experiment makes use of the period of oscillation to calculate the moment of inertia about the vertial axis. The calculation of the

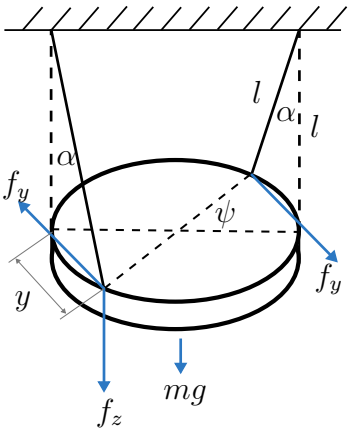


Figure A.2: Inertia experiment perturbation.

mass moment of inertia about the vertical axis is derived next. From Fig. A.2, one can see that

$$\begin{aligned} \tan\alpha &= \frac{f_y}{f_z} \\ &= \frac{f_y}{\frac{1}{2}mg}. \end{aligned} \tag{A.5}$$

Applying the small angle approximation, yields

$$\begin{aligned} f_y &= \frac{mg}{2}\tan\alpha \\ &\approx \frac{mg}{2}\alpha, \text{ and} \end{aligned} \tag{A.6}$$

$$\begin{aligned} y &= \frac{d}{2}\sin\psi \\ &\approx \frac{d}{2}\psi. \end{aligned} \tag{A.7}$$

Using Eq. (A.7), the term α is obtained as

$$\begin{aligned} \sin\alpha &= \frac{y}{l} \\ &= \frac{\frac{d}{2}\psi}{l} \\ \alpha &\approx \frac{d}{2l}\psi. \end{aligned} \tag{A.8}$$

Substituting Eq. (A.8) into Eq. (A.6), yields

$$f_y = \frac{mgd}{4l}\psi. \tag{A.9}$$

Using Newton's second law of motion and calculating the total torque of the system, yields

$$\begin{aligned} \tau &= I\ddot{\psi}, \\ f_y d &= I\ddot{\psi}, \text{ and} \\ \ddot{\psi} - \frac{mgd^2}{4Il}\psi &= 0. \end{aligned} \tag{A.10}$$

It is clear from the differential equation of Eq. (A.10), that the natural frequency, ω_n, of the system is

$$\omega_n = \sqrt{\frac{mgd^2}{4Il}}. \tag{A.11}$$

The period of the oscillations, t_p, and subsequently the mass moment of inertia about the vertical axis can now be calculated as

$$I = \frac{mgd^2t_p^2}{16\pi^2 l}, \text{ where} \tag{A.12}$$

$$t_p = \frac{2\pi}{\omega_n}. \tag{A.13}$$

Therefore, to calculate the moment of inertia about the vertical axis, the mass of the object needs to be known and l, d and t_p should be measured.

This was done for each of the axes of the quadrotor, to calculate its mass moment of inertia. The quadrotor was mounted in such a way that the vertical axis of the experiment corresponds to the desired axis of the vehicle for which the mass moment of inertia needs to be calculated. The results from the expirement is shown in Table A.2.

Inertia	d	l	**Time (s)**	**Periods**	t_p **(s)**	I (kgm^2)
I_{xx}	0.72	0.84	15	13	1.15	0.23
I_{yy}	0.715	0.81	15	13	1.15	0.235
I_{zz}	0.7	0.94	15	10	1.5	0.328

Table A.2: Mass Moment of Inertia Experiment Results

A.4 Normalized Motor Thrust vs PWM Mapping

The functions describing the input-output behaviour for each of the motors are

$$T_{M1}(x) = -3.98 \cdot 10^{-8}x^3 + 0.0002x^2 - 0.2774x + 118.77, \tag{A.14}$$
$$T_{M2}(x) = -4.14 \cdot 10^{-8}x^3 + 0.0002x^2 - 0.2858x + 121.27, \tag{A.15}$$
$$T_{M3}(x) = -4.07 \cdot 10^{-8}x^3 + 0.0002x^2 - 0.2788x + 117.76, \text{ and} \tag{A.16}$$
$$T_{M4}(x) = -3.87 \cdot 10^{-8}x^3 + 0.0002x^2 - 0.2630x + 109.84, \tag{A.17}$$

where x is pulse width of the PWM signal. The input signal is scaled to achieve normalized motor outputs around hover throttle. The incorporated scaling factors are given as

$$T_{M1_{norm}}(x) = T_{M1}(1.01316x), \tag{A.18}$$
$$T_{M2_{norm}}(x) = T_{M2}(1.00442x), \tag{A.19}$$
$$T_{M3_{norm}}(x) = T_{M3}(1.00074x), \text{ and} \tag{A.20}$$
$$T_{M4_{norm}}(x) = T_{M4}(x). \tag{A.21}$$

A.5 Calculation of the Virtual Yaw Moment Arm

A.5.1 Rotor Drag Point Force Method

This is a simplified method that assumes that the drag force acting on the rotor is a point force, as shown in Fig. A.3.

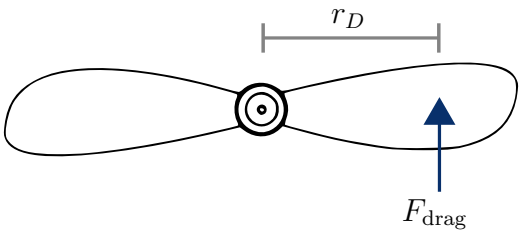


Figure A.3: Point drag force acting on a propeller.

This method is based on previous work done by [34]. Therefore, the moment about the $\boldsymbol{z_B}$ axis caused by the propeller is proportional to the distance from the hub to the point where the drag force F_{drag} is applied, denoted by r_D.

However, the propeller produces mostly lift and therefore the lift-to-drag ratio needs to be taken into account. The virtual yaw moment arm coefficient can therefore be calculated as

$$R_N = \frac{r_D}{R_{LD}}, \tag{A.22}$$

where R_{LD} is the lift-to-drag ratio.

Calculating the virtual yaw moment arm coefficient, it is assumed that $r_D = 0.18\ m$, as measured on the propeller, and that $R_{LD} = 10$, which is a typical aircraft lift-to-drag ratio. Therefore, the virtual yaw moment arm coefficient is calculated as

$$R_N = 0.018 \text{ m}. \tag{A.23}$$

A.5.2 Blade Element Theory

This method is based on work done by [40]. It makes use of blade element theory, discussed by [50], to calculate the virtual yaw moment arm coefficient.

The rotor torque and rotor thrust are defined as

$$\tau_i = \rho A C_{Ri} \left(\omega_i R\right)^2 R \text{ and} \tag{A.24}$$

$$T_i = \rho A C_{Ti} \left(\omega_i R\right)^2, \tag{A.25}$$

where ρ is the air density, A is the rotor blade area, C_{Ri} is the rotor torque coefficient of the i_{th} rotor, C_{Ti} is the rotor thrust coefficient of the i_{th} rotor, ω_i is the rotational velocity of the i_{th} rotor and R is the radius of the propeller. The rotor mechanical power output is calculated by

$$P_{mech_i} = \rho A C_{Pi} \left(\omega_i R\right)^3 = \tau_i \omega_i, \tag{A.26}$$

where C_{Pi} is the rotor power coefficient. Given Equations A.24 and A.26, it is clear that

$$C_{Ri} = C_{Pi}. \tag{A.27}$$

The rotor power coefficient is related to the rotor thrust coefficient by

$$C_{Pi} = \frac{\kappa C_{Ti}^{3/2}}{\sqrt{2}} + \frac{\sigma c_d}{8}, \tag{A.28}$$

where κ is a constant that incorporates rotor blade tip losses, c_d is the rotor blade drag coefficient and σ is the rotor solidity ratio. These values are obtained by

$$\kappa = \frac{1.13}{B}, \tag{A.29}$$

$$c_d = 0.0087 - 0.0216\alpha + 0.4\alpha^2, \text{ and} \tag{A.30}$$

$$\sigma = \frac{N\bar{c}}{\pi R}, \tag{A.31}$$

where

$$B = 1 - \frac{\sqrt{2C_{Ti}}}{N}. \tag{A.32}$$

In these equations, N is the number of blades on the rotor, α is the rotor disk angle of attack and $\bar{c}$ is the mean rotor chord length. The mean chord length is the mean length of the airfoil, illustrated in Fig. A.4.

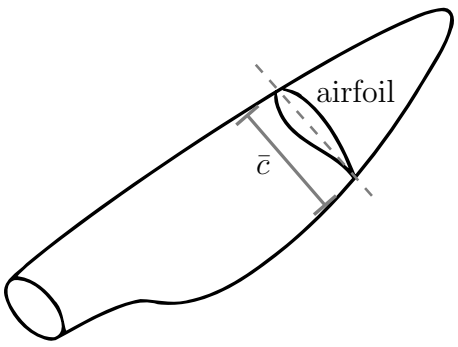


Figure A.4: Rotor airfoil illustration.

Using the equations above, the virtual yaw moment arm coefficient can be calculated. It is calculated at hover, meaning that the thrust and rotational velocity values are those at hover throttle. The hover thrust is obtained from the thrust test jig used to determine the thrust time constant and the hover rotational speed is measured with a tacometer. At hover, the angle of attack of the vehicle is zero and therefore $\alpha = 0$. The following measurements are available from the physical system Using these values and the equations mentioned above, the following

Variable	**Measurement**
N	2
R	0.2286 m
$A = \pi R^2$	0.1642 m^2
$\bar{c}$	0.035 m
T_{hover}	11.03625 N
ω_{hover}	295.92 rad/s

calculations are made to obtain the virtual yaw moment arm.

First, the rotor thrust coefficient is calculated as

$$C_{Ti} = 0.012. \tag{A.33}$$

The rotor power coefficient, which is equal to the rotor torque coefficient, is calculated next. The needed parameters are first calculated by

$$B = 1 - \frac{\sqrt{2C_{Ti}}}{N} \tag{A.34}$$
$$= 0.923, \tag{A.35}$$
$$\kappa = \frac{1.13}{B} \tag{A.36}$$
$$= 1.225, \tag{A.37}$$
$$c_d = 0.0087, \text{ and} \tag{A.38}$$
$$\sigma = \frac{N\bar{c}}{\pi R} \tag{A.39}$$
$$= 0.0975. \tag{A.40}$$

The rotor torque coefficient is calculated as

$$C_{Pi} = \frac{\kappa C_{Ti}^{3/2}}{\sqrt{2}} + \frac{\sigma c_d}{8}$$
$$= 0.001243.$$

The torque of each rotor at hover is calculated as

$$\tau_i = 0.262.$$

Finally, given the torque and thrust at hover, the virtual yaw moment arm coefficient is obtained by

$$R_N = \frac{\tau_i}{T_i} = \frac{0.262}{11.03625} = 0.0237 \text{ m}. \tag{A.41}$$

A.5.3 Experimental Method

This method makes use of thrust and torque measurements during an experiment to calculate the virtual yaw moment arm coefficient. Such experimental data is available by the manufacturer of the propulsion system and is given in Table A.1. The thrust and torque values in this table are used with Eq. (4.1) to calculate the virtual yaw moment arm coefficient. The coefficient is determined for each throttle value and is shown in Table A.3.

Throttle	**Torque (Nm)**	**Thrust (kg)**	**Virtual Yaw Moment Arm (m)**
50%	0.290	1.318	0.0224
55%	0.344	1.612	0.0218
60%	0.411	1.901	0.0220
65%	0.472	2.259	0.0213
75%	0.605	2.835	0.0218
85%	0.737	3.477	0.0216
100%	0.918	4.355	0.0215

Table A.3: Virtual yaw moment arm experimental calculations.

The calculated coefficient for each of the throttle values averages to

$$R_N = 0.0218 \text{ m}. \tag{A.42}$$

B. Quadrotor Control System Design

B.1 Calculations of the Force to Attitude and Thrust Conversion

The force commanded from the inertial linear velocity controller is equivalent to the thrust vector of the vehicle in the body frame, as shown in Fig. B.1.

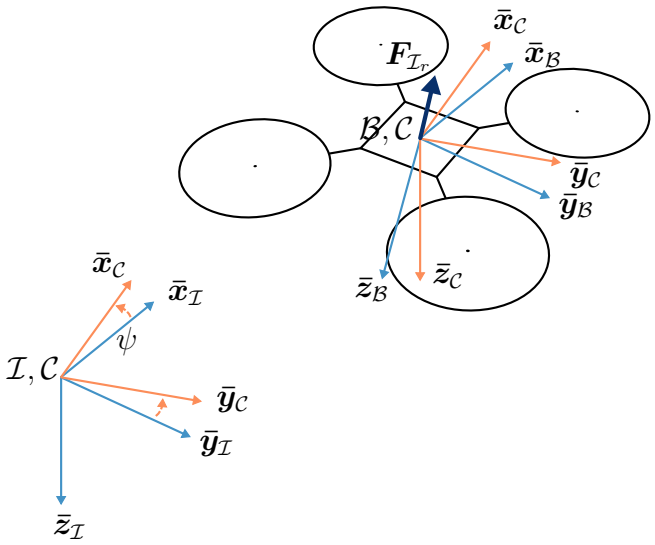


Figure B.1: Illustration of the commanded inertial force.

The commanded force is transformed to a desired attitude and thrust that will allow the vehicle to follow the reference velocity. This transformation is described by [45]. The authors calculate the desired attitude, denoted by $\bar{\boldsymbol{x}}_{\mathcal{B}_r}$, $\bar{\boldsymbol{y}}_{\mathcal{B}_r}$ and $\bar{\boldsymbol{z}}_{\mathcal{B}_r}$, from the desired force $\boldsymbol{F}_{\mathcal{I}_r}$ as

$$\bar{\boldsymbol{z}}_{\mathcal{B}_r} = -\frac{\boldsymbol{F}_{\mathcal{I}_r}}{|\boldsymbol{F}_{\mathcal{I}_r}|}, \tag{B.1}$$

$$\bar{\boldsymbol{y}}_{\mathcal{C}_r} = \begin{bmatrix} -\sin\psi_r & \cos\psi_r & 0 \end{bmatrix}^T, \tag{B.2}$$

$$\bar{\boldsymbol{x}}_{\mathcal{B}_r} = \frac{\bar{\boldsymbol{y}}_{\mathcal{C}_r} \times \bar{\boldsymbol{z}}_{\mathcal{B}_r}}{|\bar{\boldsymbol{y}}_{\mathcal{C}_r} \times \bar{\boldsymbol{z}}_{\mathcal{B}_r}|}, \text{ and} \tag{B.3}$$

$$\bar{\boldsymbol{y}}_{\mathcal{B}_r} = \bar{\boldsymbol{z}}_{\mathcal{B}_r} \times \bar{\boldsymbol{x}}_{\mathcal{B}_r}, \tag{B.4}$$

where $\mathcal{C}$ is an intermediate frame described by rotating the inertial frame about the $\bar{\boldsymbol{z}}_{\mathcal{I}}$ axis by the yaw angle, ψ_r. The commanded inertial force is described by the vector

$$\boldsymbol{F}_{\mathcal{I}_r} = \begin{bmatrix} F_{\mathcal{I}_{N,r}} & F_{\mathcal{I}_{E,r}} & F_{\mathcal{I}_{D,r}} \end{bmatrix}^T. \tag{B.5}$$

The rotation matrix is then described by

$$\boldsymbol{R}_v = \begin{bmatrix} \bar{\boldsymbol{x}}_{\mathcal{B}_r} & \bar{\boldsymbol{y}}_{\mathcal{B}_r} & \bar{\boldsymbol{z}}_{\mathcal{B}_r} \end{bmatrix}^T. \tag{B.6}$$

The desired quaternion is calculated from this rotation matrix and passed to the angle controllers.

B.2 Linear Plant of the Longitudinal Velocity

Consider the case where a quadrotor is flying at a constant longitudinal velocity and height, as shown in Fig. B.2. Taking the sum of the forces in the vertical axis, yields

$$-F_q \cos(-\theta) + mg = 0 \tag{B.7}$$

$$F_q \cos\theta = mg, \tag{B.8}$$

where m is the mass of the quadrotor and g is the gravitational acceleration constant. Applying small angle approximation yields

$$F_q \approx mg. \tag{B.9}$$

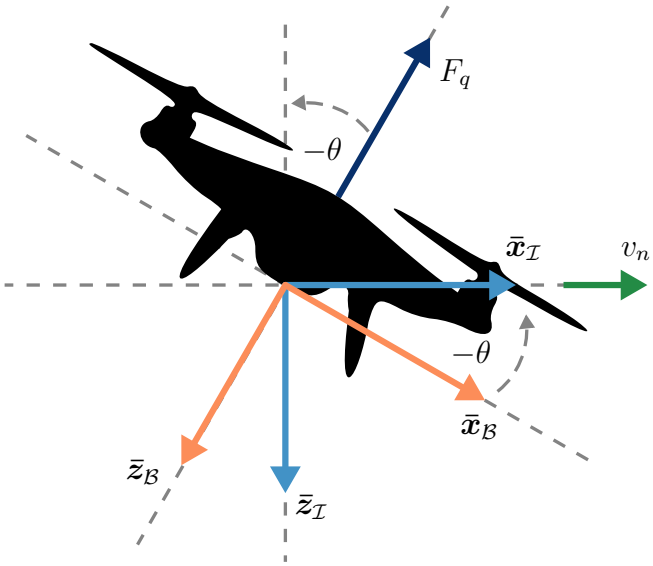


Figure B.2: Illustration of the inertial linear velocity dynamics of a quadrotor.

The longitudinal inertial force acting in on the vehicle is

$$F_{\mathcal{I}_N} = F_q \sin(-\theta). \tag{B.10}$$

Applying the small angle approximation and substituting Eq. (B.9), yields

$$F_{\mathcal{I}_N} \approx -mg\theta. \tag{B.11}$$

Transforming the rotation of the angle $-\theta$ about the $\bar{\boldsymbol{y}}_{\mathcal{I}}$ axis to a quaternion, is done by

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\left(-\frac{\theta}{2}\right) \\ 0 \cdot \sin\left(-\frac{\theta}{2}\right) \\ 1 \cdot \sin\left(-\frac{\theta}{2}\right) \\ 0 \cdot \sin\left(-\frac{\theta}{2}\right) \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ 0 \\ -\sin\left(\frac{\theta}{2}\right) \\ 0 \end{bmatrix}. \tag{B.12}$$

Applying the small angle approximation, yields

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -\frac{\theta}{2} \\ 0 \end{bmatrix}. \tag{B.13}$$

The pitch angle is calculated as

$$q_2 = -\frac{\theta}{2} \tag{B.14}$$

$$\theta = -2q_2. \tag{B.15}$$

The longitudinal inertial force now becomes

$$F_{\mathcal{I}_N} = 2mgq_2. \tag{B.16}$$

The longitudinal dynamics is obtained by making use of Newton's second of law motion, yielding

$$F_{\mathcal{I}_N} = m\dot{V}_N. \tag{B.17}$$

The linear plant is then calculated as

$$G_{V_N}(s) = \frac{V_N(s)}{q_2(s)} = \frac{2g}{s}. \tag{B.18}$$

B.3 Lateral, Heave and Directional Controller Design

The design of the longitudinal control system is described in detail in Section 6.1. In this section, the design process of the lateral, heave and directional control systems are briefly described. The design process follows much of the same structure as in Section 6.1. Therefore, a complete control system design is not given to avoid duplication. Rather, the control gains and control parameters are given in Appendix B.4.

The lateral control system has the exact same structure as the longitudinal control system, described in Section 6.1. The control gains of the angular rate controller differs, as the mass moment of inertia about the x-axis is slightly different of that about the y-axis. The roll angle, lateral linear velocity and lateral position controller has the exact same control gains as the longitudinal counterparts.

The heave controllers consist of the linear velocity and position controllers in the vertical axis. They are responsible for the height control of the vehicle. These controllers have the same structure as that of the longitudinal linear velocity and position controllers. The same design process is followed.

The directional controllers consist of the yaw rate and yaw angle controllers. These controllers have the same structure as that of the pitch rate and pitch controllers. The same design process is followed.

B.4 Control System Gains

In this section, the designed flight control system gains and parameters of the quadrotor are given.

B.4.1 Attitude Controllers

The controller gains and parameters regarding the attitude controllers are given in this section. This includes the PID gains, the maximum and minimum commanded control references and

the filter cutoff frequencies. The IMU data is also filtered before processed by the EKF. The filter parameters are given by Table B.1.

Parameter	Value
Gyroscope filter cutoff frequency	30 Hz
Acceloremeter filter cutoff frequency	30 Hz

Table B.1: The IMU filter parameters.

The IMU filter cutoff frequencies are quite low as the large motor-propeller pairs induce vibrations. The low cutoff frequencies produces signals with very low noise. Therefore, the D-term LPFs of the angular rate controllers are disabled. A disabled filter is denoted by a ∞ Hz cutoff frequency in the rest of the section.

Pitch Rate

The controller gains and corresponding parameters relating to the pitch rate controller is given in Table B.2.

Parameter	Value
P Gain	0.086
I Gain	0.020
D Gain	0.003
D-term LPF Cutoff Frequency	∞ Hz
Maximum Commanded Pitch Rate	120 deg/s
Minimum Commanded Pitch Rate	-120 deg/s

Table B.2: The pitch rate controller gains and corresponding parameters.

Pitch Angle

The controller gains and corresponding parameters relating to the pitch angle controller is given in Table B.3.

Parameter	Value
P Gain	3
Maximum Commanded Pitch Angle	30 deg
Minimum Commanded Pitch Angle	-30 deg

Table B.3: The pitch angle controller gains and corresponding parameters.

Roll Rate

The controller gains and corresponding parameters relating to the roll rate controller is given in Table B.4.

Parameter	Value
P Gain	0.0842
I Gain	0.0196
D Gain	0.0029
D-term LPF Cutoff Frequency	∞ Hz
Maximum Commanded Roll Rate	120 deg/s
Minimum Commanded Roll Rate	-120 deg/s

Table B.4: The roll rate controller gains and corresponding parameters.

Roll Angle

The controller gains and corresponding parameters relating to the roll angle controller is given in Table B.5.

Parameter	Value
P Gain	3
Maximum Commanded Roll Angle	30 deg
Minimum Commanded Roll Angle	-30 deg

Table B.5: The roll angle controller gains and corresponding parameters.

Yaw Rate

The controller gains and corresponding parameters relating to the yaw rate controller is given in Table B.6.

Parameter	Value
P Gain	0.035
I Gain	0.03
D Gain	0.0
D-term LPF Cutoff Frequency	∞ Hz
Maximum Commanded Yaw Rate	45 deg/s
Minimum Commanded Yaw Rate	-45 deg/s

Table B.6: The yaw rate controller gains and corresponding parameters.

Yaw Angle

The controller gains and corresponding parameters relating to the yaw angle controller is given in Table B.7.

Parameter	Value
P Gain	1.2

Table B.7: The yaw angle controller gains and corresponding parameters.

B.4.2 Translational Controllers

The controller gains and parameters regarding the translational controllers are given in this section. This includes the PID gains, the maximum and minimum commanded control references and the filter cutoff frequencies.

Longitudinal Velocity

The controller gains and corresponding parameters relating to the longitudinal velocity controller is given in Table B.8.

Parameter	Value
P Gain	0.048
I Gain	0.008
D Gain	0.002
D-term LPF Cutoff Frequency	5 Hz
Maximum Commanded Velocity	12 m/s
Minimum Commanded Velocity	−12 m/s

Table B.8: The longitudinal velocity controller gains and corresponding parameters.

North Position

The controller gains and corresponding parameters relating to the north position controller is given in Table B.9.

Parameter	Value
P Gain	0.35

Table B.9: The north position controller gains and corresponding parameters.

Lateral Velocity

The controller gains and corresponding parameters relating to the lateral velocity controller is given in Table B.10.

Parameter	Value
P Gain	0.048
I Gain	0.008
D Gain	0.002
D-term LPF Cutoff Frequency	5 Hz
Maximum Commanded Velocity	12 m/s
Minimum Commanded Velocity	−12 m/s

Table B.10: The lateral velocity controller gains and corresponding parameters.

East Position

The controller gains and corresponding parameters relating to the east position controller is given in Table B.11.

Parameter	Value
P Gain	0.35

Table B.11: The east position controller gains and corresponding parameters.

Down Velocity

The controller gains and corresponding parameters relating to the downward velocity controller is given in Table B.12.

Parameter	Value
P Gain	0.1
I Gain	0.01
D Gain	0.0
D-term LPF Cutoff Frequency	5 Hz
Maximum Commanded Velocity	1 m/s
Minimum Commanded Velocity	-3 m/s

Table B.12: The downward velocity controller gains and corresponding parameters.

Down Position

The controller gains and corresponding parameters relating to the downward position controller is given in Table B.13.

Parameter	Value
P Gain	0.9

Table B.13: The downward position controller gains and corresponding parameters.